MPC5746C Reference Manual

Supports MPC5744B, MPC5745B, MPC5746B, MPC5744C, MPC5745C and MPC5746C

Document Number: MPC5746CRM
Rev. 4, 12/2016



Contents

Section number **Title** **Page**

Chapter 1
About This Manual

1.1 Audience 95

1.2 Organization 95

1.3 Module descriptions 95

1.3.1 Example: chip-specific information that clarifies content in the same chapter 96

1.3.2 Example: chip-specific information that refers to a different chapter 97

1.4 Register descriptions 98

1.5 Conventions 99

1.5.1 Notes, Cautions, and Warnings 99

1.5.2 Numbering systems 99

1.5.3 Typographic notation 100

1.5.4 Special terms 100

Chapter 2
Introduction

2.1 Introduction 103

2.1.1 Target Applications 103

2.2 Block diagram 103

2.3 Family comparison 104

2.4 Feature list 108

2.5 Package Options 110

2.6 Modes of Operation 110

Chapter 3
Memory Map

3.1 Introduction 115

3.2 System memory map 115

3.3 NVM memory map 116

3.4 UTEST Memory Map 118

3.5 RAM memory map ... 119
3.6 Peripheral Bridges A and B memory maps ... 120
3.6.1 Peripheral Bridge A memory map ... 120
3.6.2 Peripheral Bridge B memory map ... 124
3.7 MC_CGM mapped peripherals memory map ... 125
3.8 PCU mapped peripherals memory map ... 125
3.9 DMAMUX mapped peripherals memory map ... 126

Chapter 4
Signal Description

4.1 Generic pins/balls ... 127
4.1.1 MSCR assignments ... 127
4.2 PAD keeping feature in LPU/STANDBY mode ... 128

Chapter 5
Security Overview

5.1 Introduction ... 133
5.2 Basic security ... 133
5.3 Advanced security ... 133
5.4 Detailed security information ... 134

Chapter 6
Platform Overview

6.1 Introduction ... 135
6.2 User/Privilege Access ... 136

Chapter 7
Power Management Overview

7.1 Introduction ... 139
7.1.1 Features ... 140
7.2 Power domains ... 141
7.2.1 Power Domain0 (PD0) ... 142
7.2.2 Power Domain1 (PD1) ... 142
7.2.3 Power Domain2 (PD2) ... 142

7.2.4 Flash HV power domain ... 143
7.2.5 SRAM domains ... 143
7.2.6 SRAM Configuration ... 143
7.3 User operating modes, power domains, and clock gating ... 144
7.4 Voltage regulators ... 146
7.4.1 FPREG voltage regulator ... 146
7.4.2 Internal regulation (Internal ballast) ... 148
7.4.3 LPREG voltage regulator ... 148
7.4.4 ULPREG voltage regulator ... 148
7.4.5 Flash voltage regulator ... 149
7.4.6 Regulation mode decoding scheme ... 149
7.4.7 Voltage monitoring ... 150
7.4.8 Applications and Low Power Modes ... 152
7.5 External Regulation ... 152
7.5.1 PORST in EXTERNAL REGULATION ... 152
7.5.2 External regulator contol ... 153
7.6 Accelerated Low Power Exit ... 154

Chapter 8
Low Power Subsystem (LPU)

8.1 Introduction ... 157
8.2 Block diagram ... 157
8.3 Features ... 158
8.4 Modes of operation ... 160
8.5 Transition to LPU mode ... 161
8.6 Exit from LPU Modes ... 165
8.7 Clocking ... 166
8.7.1 LPU System Clock Generation ... 166
8.7.2 FlexCAN0 Clocking ... 168
8.7.3 LinFlexD Clocking ... 168

8.7.4 SPI_0 Clocking ... 168
8.7.5 LPU clock source control ... 169
8.8 Reset ... 169
8.9 Interrupts ... 170
8.10 Boot ... 171
8.11 LPU Controller (LPU_CTL) ... 171
8.11.1 Memory map and register definition ... 171
8.11.2 LPU Mode Control Register (LPU_MCTL) ... 172
8.11.3 LPU Run Configuration Register (LPU_RUN_CF) ... 174
8.11.4 LPU Stop Configuration Register (LPU_STOP_CF) ... 176
8.11.5 LPU Standby Configuration Register (LPU_STANDBY_CF) ... 177
8.11.6 LPU Status Register (LPU_SR) ... 178
8.11.7 LPU Interrupt control register (LPU_ICR) ... 181
8.11.8 Module Disable (LPU_MDIS) ... 183

Chapter 9
Clocking Overview

9.1 Introduction ... 185
9.2 System clock generation ... 186
9.3 Dynamic Frequency Scaling ... 187
9.3.1 Overview ... 187
9.3.2 Linear DFS Configuration ... 189
9.3.3 Non-Linear DFS Configuration ... 190
9.3.4 DFS Limitations ... 191
9.3.5 NLDFS Abort on Interrupt ... 191
9.4 Peripheral clock generation ... 191
9.4.1 Overview ... 191
9.4.2 Enabling/Disabling of peripherals and clock sources ... 192
9.4.3 Communication interfaces ... 192
9.4.4 Timers Clock ... 199

9.4.5 ADC/BCTU/eMIOS clocking....202
9.4.6 CLKOUT clocking....202
9.4.7 HSM clocking....204
9.4.8 LPU clocking....204
9.5 MC_CGM registers....204
9.6 Built-In Self-Test clocking....205
9.7 System Clock Distribution....205
9.7.1 JTAG Frequencies....207
9.7.2 Peripheral Clock Restrictions....208
9.8 Default clock configuration....208
9.9 Clock sources....209
9.9.1 PLL....209
9.9.2 Fast External Oscillator (FXOSC)....211
9.9.3 Slow external oscillator (SXOSC)....212
9.9.4 16 MHz internal RC oscillator (FIRC)....213
9.9.5 128 kHz internal RC (SIRC)....214
9.10 Clock monitoring....215
9.10.1 CMU configuration....215
9.11 Loss of system clock behavior....218
9.11.1 Loss of PLL lock/FXOSC clock....218
9.11.2 Loss of FIRC clock....218

Chapter 10
Embedded Memories

10.1 Overview....221
10.2 System SRAM....221
10.2.1 SRAM Controller (PRAMC)....222
10.2.2 Overlay SRAM....223
10.3 Flash memory....224
10.3.1 Flash memory controller....225

10.3.2 Flash memory array ... 226
10.3.3 Flash organization ... 227
10.3.4 UTest NVM memory space ... 228
10.3.5 Overlay flash memory ... 229
10.4 End-to-end Error Correction Code (e2eECC) ... 229
10.5 Security features ... 232

Chapter 11
Device Configuration Format (DCF) records

11.1 Introduction ... 233
11.2 DCF clients ... 234
11.2.1 Safety features of DCF clients ... 234
11.2.2 DCF client modification rules ... 235
11.3 DCF record structure ... 236
11.4 DCF records sequence ... 237
11.5 Chip configuration records ... 239
11.5.1 External Regulation Exit Trim Wait (PMCDIG_ERETW) ... 239
11.5.2 Security and serial boot mode records ... 240
11.5.3 STCU configuration records ... 240

Chapter 12
Debug Overview

12.1 Introduction ... 241
12.2 Features ... 243
12.3 Debug scheme ... 244
12.4 Modes of operation ... 245
12.4.1 Nexus Reset Mode ... 245
12.4.2 Full-port Mode ... 246
12.4.3 Reduced-port Mode ... 246
12.4.4 Disabled-Port Mode ... 246
12.4.5 Multi mode nexus TAP ... 246

12.5 External signal description....247
12.6 Memory Map and Registers....248
12.7 Functional Description....248
12.7.1 Enabling Nexus Clients for TAP Access....248
12.7.2 TAP Sharing....248
12.7.3 Configuring the NDI for Nexus Messaging....249
12.7.4 TCK clock relation with NPC clock....249
12.7.5 Programmable MCKO frequency....249
12.7.6 Cross Triggering Control....251
12.7.7 Time Stamping....252
12.8 Nexus Port Controller (NPC)....253
12.8.1 NPC Handshake module....253
12.9 e200z4 core and e200z2 core debug support....258
12.10 JTAG Controller (JTAGC)....258
12.11 JTAG ID....258
12.12 CJTAGC....259
12.13 JDC....259
12.14 Debug implementation....259
12.14.1 Debug Capabilities....259
12.14.2 Debug port....260
12.15 Nexus Class 3+ Auxiliary Port....260
12.16 e200z2 Nexus 3....261
12.17 e200z4 Nexus 3....262
12.18 Nexus Crossbar Multi-master Client (NXMC)....262

Chapter 13
Reset and Boot Overview

13.1 Introduction....265
13.1.1 Global Reset....265
13.1.2 Local Reset....265

13.1.3 Reset sources....265
13.2 Global Reset Process....268
13.2.1 Overview....268
13.2.2 Reset Process Modules....269
13.2.3 Reset Process Sequences....271
13.2.4 Reset Process State Transitions....272
13.2.5 Module Status During Reset Process....282
13.3 Boot....283
13.3.1 Power Up....283
13.3.2 Exit from STANDBY/LPU_STANDBY to DRUN....283
13.3.3 Exit from LPU STANDBY to LPU_RUN....284
13.3.4 Entry from RUNx/DRUN to LPU_RUN....284
13.3.5 Interrupt Vector Prefix retention during STANDBY....284

Chapter 14
Functional Safety

14.1 Introduction....285

Chapter 15
System Integration Unit Lite2 (SIUL2)

15.1 Introduction....287
15.1.1 Overview....287
15.1.2 Features....289
15.1.3 Register protection....290
15.2 Memory map and register description....291
15.2.1 SIUL2 MCU ID Register #1 (SIUL2_MIDR1)....338
15.2.2 SIUL2 MCU ID Register #2 (SIUL2_MIDR2)....339
15.2.3 SIUL2 Interrupt Status Flag Register0 (SIUL2_ISR0)....340
15.2.4 SIUL2 Interrupt Request Enable Register0 (SIUL2_IRER0)....344
15.2.5 SIUL2 Interrupt Request Select Register0 (SIUL2_IRSR0)....347
15.2.6 SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)....350

15.2.7 SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)..... 353
15.2.8 SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)..... 356
15.2.9 SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR*n*)..... 358
15.2.10 SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)..... 359
15.2.11 SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR*n*)..... 360
15.2.12 SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR*n*)..... 362
15.2.13 SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO*n*)..... 365
15.2.14 SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI*n*)..... 366
15.2.15 SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO*n*)..... 368
15.2.16 SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI*n*)..... 369
15.2.17 SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO*n*)..... 369
15.3 Functional description..... 370
15.3.1 General..... 370
15.3.2 Pad Control..... 370
15.3.3 General Purpose Input and Output Pads (GPIO)..... 371
15.3.4 External Interrupts Requests (REQ Pins)..... 372

Chapter 16
Crossbar Switch (AXBS)

16.1 Chip-specific AXBS information..... 377
16.1.1 Crossbar Switch configuration..... 377
16.1.2 AXBS 0 Master and Slave assignments..... 378
16.1.3 PRS register reset values..... 379
16.1.4 CRS register reset values..... 379
16.2 Introduction..... 380
16.2.1 Features..... 380
16.3 Memory Map / Register Definition..... 380
16.3.1 Priority Registers Slave (AXBS_PRS*n*)..... 382
16.3.2 Control Register (AXBS_CRS*n*)..... 385
16.3.3 Master General Purpose Control Register (AXBS_MGPCR*n*)..... 387

16.4 Functional Description....388
16.4.1 General operation....388
16.4.2 Register coherency....389
16.4.3 Arbitration....389
16.5 Initialization/application information....392

Chapter 17
Peripheral Bridge

17.1 Chip-specific PBridge information....395
17.1.1 PBRIDGE Configuration....395
17.1.2 Register reset values....395
17.2 Introduction....399
17.2.1 Features....399
17.2.2 General operation....399
17.3 Memory map/register definition....400
17.3.1 Master Privilege Register A (AIPS_MPRA)....402
17.3.2 Master Privilege Register B (AIPS_MPRB)....405
17.3.3 Peripheral Access Control Register (AIPS_PACR*n*)....407
17.3.4 Off-Platform Peripheral Access Control Register (AIPS_OPACR*n*)....412
17.4 Functional description....418
17.4.1 Access support....418

Chapter 18
Decorated Storage Memory Controller (DSMC)

18.1 Introduction....419
18.2 Decorated stores: st[b,h,w]d{cb}x rS,rA,rB....421
18.2.1 Bit Field Insert (BFINS) into an 8, 16 or 32-bit memory container....422
18.3 DSMC instantiations....426

Chapter 19
Intelligent AHB Gasket (IAHBG)

19.1 Chip-specific IAHBG information....429
19.1.1 IAHBG configuration....429

19.1.2 IAHBG clock selection ratios..... 429
19.1.3 IAHBG features enabled by Platform Configuration Module (PCM)..... 430
19.2 Introduction..... 430
19.3 Timing modes..... 430
19.3.1 1:1..... 431
19.3.2 2:1..... 431
19.3.3 1:2..... 431

Chapter 20
System Memory Protection Unit (SMPU)

20.1 Chip-specific SMPU information..... 433
20.1.1 System Memory Protection Unit (SMPU) configuration..... 433
20.2 Overview..... 435
20.2.1 Block diagram..... 435
20.2.2 Features..... 436
20.3 Memory map/register definition..... 436
20.3.1 Control/Error Status Register 0 (SMPU_CES0)..... 442
20.3.2 Control/Error Status Register 1 (SMPU_CES1)..... 443
20.3.3 Error Address Register, Bus Master (SMPU_ERR_ADR*n*)..... 443
20.3.4 Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL*n*)..... 444
20.3.5 Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL*n*)..... 445
20.3.6 Region Descriptor n, Word 0 (SMPU_RGD*n*_WRD0)..... 446
20.3.7 Region Descriptor n, Word 1 (SMPU_RGD*n*_WRD1)..... 447
20.3.8 Region Descriptor n, Word 2 (SMPU_RD_*n*W2)..... 447
20.3.9 Region Descriptor n, Word 3 (SMPU_RGD*n*_WRD3)..... 449
20.3.10 Region Descriptor n, Word 4 (SMPU_RGD*n*_WRD4)..... 450
20.3.11 Region Descriptor n, Word 5 (SMPU_RGD*n*_WRD5)..... 451
20.4 Functional description..... 452
20.4.1 Access evaluation macro..... 452
20.4.2 Putting it all together and error terminations..... 454

20.5 Initialization information..... 455
20.6 Application information.....455

Chapter 21
Semaphores2 (SEMA42)

21.1 Introduction.....457
21.1.1 Features..... 457
21.2 Memory map/register definition..... 459
21.2.1 Gate Register (SEMA42_GATE*n*)..... 459
21.2.2 Reset Gate Write (SEMA42_RSTGT_W).....460
21.2.3 Reset Gate Read (SEMA42_RSTGT_R).....462
21.3 Functional description.....463

Chapter 22
Platform Configuration Module (PCM)

22.1 Memory map and register definition.....465
22.1.1 Bus Bridge Configuration Register 0 (PCM_IAHB_BE0).....466
22.1.2 Bus Bridge Configuration Register 1 (PCM_IAHB_BE1).....468
22.1.3 IPS Clock Gating Module Enable 0 Register (PCM_IPS_CGM_EN0)..... 470

Chapter 23
Interrupt Controller (INTC)

23.1 Chip-specific INTC information.....473
23.1.1 INTC configuration.....473
23.1.2 INTC interrupt sources.....473
23.1.3 INTC implemented registers.....485
23.1.4 Interrupts handling during mode transitions..... 486
23.2 Introduction.....486
23.3 Block diagram.....487
23.4 Features.....488
23.5 Modes of operation..... 489
23.5.1 Software vector mode..... 489
23.5.2 Hardware vector mode.....490

23.6 Memory map and register definition....490
23.6.1 INTC Block Configuration Register (INTC_BCR)....534
23.6.2 INTC Master Protection Register (INTC_MPROT)....535
23.6.3 INTC Current Priority Register for Processor 0 (INTC_CPR0)....536
23.6.4 INTC Current Priority Register for Processor 1 (INTC_CPR1)....537
23.6.5 INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)....538
23.6.6 INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)....539
23.6.7 INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)....540
23.6.8 INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)....541
23.6.9 INTC Software Set/Clear Interrupt Register (INTC_SSCIR*n*)....541
23.6.10 INTC Priority Select Register (INTC_PSR*n*)....543
23.7 Functional description....544
23.7.1 Interrupt request sources....544
23.7.2 Priority management....546
23.7.3 Handshaking with processor....548
23.8 Initialization/application information....551
23.8.1 Initialization flow....551
23.8.2 Interrupt exception handler....552
23.8.3 ISR, RTOS, and task hierarchy....553
23.8.4 Order of execution....554
23.8.5 Priority ceiling protocol....555
23.8.6 Selecting priorities according to request rates and deadlines....558
23.8.7 Software-settable interrupt requests....559
23.8.8 Lowering priority within an ISR....560
23.8.9 Negating an interrupt request outside of its ISR....561
23.8.10 Examining LIFO contents....561
23.9 Interrupt sources....562

Chapter 24
Wakeup Unit (WKPU)

24.1 Chip-specific WKPU information ... 563
24.1.1 Wakeup configuration ... 563
24.1.2 NMI configuration ... 563
24.1.3 Interrupts ... 564
24.1.4 Wakeup Sources for various modes ... 566
24.1.5 NMI on standby exit ... 566
24.1.6 Register reset values ... 567
24.1.7 Pull control of wakeup pads ... 567
24.2 Introduction ... 568
24.3 Features ... 569
24.4 External signal description ... 570
24.5 WKPU memory map and register definition ... 570
24.5.1 NMI Status Flag Register (WKPU_NSR) ... 570
24.5.2 NMI Configuration Register (WKPU_NCR) ... 572
24.5.3 Wakeup/Interrupt Status Flag Register (WKPU_WISR) ... 575
24.5.4 Interrupt Request Enable Register (WKPU_IRER) ... 576
24.5.5 Wakeup Request Enable Register (WKPU_WRER) ... 577
24.5.6 Wakeup/Interrupt Rising-Edge Event Enable Register (WKPU_WIREER) ... 577
24.5.7 Wakeup/Interrupt Falling-Edge Event Enable Register (WKPU_WIFEER) ... 578
24.5.8 Wakeup/Interrupt Filter Enable Register (WKPU_WIFER) ... 578
24.5.9 Wakeup/Interrupt Pull Enable Register (WKPU_WIPER) ... 579
24.6 Functional description ... 579
24.6.1 Non-maskable interrupts ... 579
24.7 Initialization Information ... 581
24.7.1 Glitch Filter and Pad Configuration ... 581
24.7.2 Non-Maskable Interrupts ... 581
24.7.3 Reset Request ... 582

Chapter 25
PLL Digital Interface (PLLDIG)

25.1 Introduction....583
25.2 Block Diagram....583
25.3 Features....583
25.4 Modes of operation....584
25.4.1 Normal mode with reference, PLL enabled....584
25.5 Memory map and register definition....584
25.5.1 PLL Calibration Register 3 (PLLDIG_PLLCAL3)....585
25.5.2 PLLDIG PLL Control Register (PLLDIG_PLLCR)....586
25.5.3 PLLDIG PLL Status Register (PLLDIG_PLLSR)....588
25.5.4 PLLDIG PLL Divider Register (PLLDIG_PLLDV)....590
25.5.5 PLLDIG PLL Frequency Modulation Register (PLLDIG_PLLFM)....592
25.5.6 PLLDIG PLL Fractional Divide Register (PLLDIG_PLLFD)....594
25.6 Functional description....595
25.6.1 Input clock frequency....596
25.6.2 Clock configuration....596
25.6.3 Fault Collection and Control Unit (FCCU)....597
25.6.4 Frequency modulation....597
25.6.5 PLL Wakeup from low power mode....598
25.6.6 Calibration feature....598
25.7 Initialization information....599

Chapter 26
Slow Internal 128 kHz Oscillator (SIRC)

26.1 Introduction....601
26.2 Features....601
26.3 Memory map and register definition....601
26.3.1 Control register (SIRC_CTL)....601
26.4 Functional description....603

Chapter 27
Fast Internal 16 MHz RC Oscillator (FIRC)

27.1 Introduction....605
27.2 Features....605
27.3 Memory map and register definition....605
27.3.1 Control register (FIRC_CTL)....606
27.4 Functional description....607

Chapter 28
Slow External 32 KHz Oscillator (SXOSC)

28.1 Introduction....609
28.2 Features....609
28.3 Memory map and register definition....609
28.3.1 SXOSC_CTL....610
28.4 Functional description....611

Chapter 29
Fast OSC Digital Interface (FXOSC)

29.1 Introduction....613
29.2 Functional description....613
29.2.1 Oscillator power-down control and status....614
29.2.2 Oscillator startup delay....614
29.2.3 Oscillator clock available interrupt....614
29.2.4 Oscillator bypass mode....615
29.2.5 Crystal oscillator mode selection....615
29.2.6 Oscillator divider....616
29.3 Memory map and register definition....617
29.3.1 FXOSC Control Register (FXOSC_CTL)....617

Chapter 30
Clock Monitor Unit (CMU)

30.1 Introduction....621
30.1.1 Main features....622
30.2 Block diagram....622
30.3 Signals....622

30.4 Register description and memory map..... 623
30.4.1 CMU Control Status Register (CMU_CSR)..... 624
30.4.2 CMU Frequency Display Register (CMU_FDR)..... 625
30.4.3 CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)..... 626
30.4.4 CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)..... 626
30.4.5 CMU Interrupt Status Register (CMU_ISR)..... 627
30.4.6 CMU Measurement Duration Register (CMU_MDR)..... 629
30.5 Functional description..... 629
30.5.1 Frequency meter..... 629
30.5.2 CLKMN0_RMT supervisor..... 630
30.5.3 CLKMN1 supervisor..... 630

Chapter 31
Clock Generation Module (MC_CGM)

31.1 Introduction..... 633
31.1.1 Overview..... 633
31.1.2 Features..... 634
31.2 External Signal Description..... 635
31.3 Memory Map and Register Definition..... 635
31.3.1 Clockout 1 Select Register (MC_CGM_CLKOUT1_SC)..... 637
31.3.2 CLKOUT1_DC Configuration Register (MC_CGM_CLKOUT1_DC)..... 638
31.3.3 Auxiliary Clock 5 Select Control Register (MC_CGM_AC5_SC)..... 639
31.3.4 Auxiliary Clock 5 Select Status Register (MC_CGM_AC5_SS)..... 640
31.3.5 System Clock Divider 6 Configuration Register (MC_CGM_SC_DC6)..... 641
31.3.6 Auxiliary Clock 8 Select Control Register (MC_CGM_AC8_SC)..... 641
31.3.7 Auxiliary Clock 8 Select Status Register (MC_CGM_AC8_SS)..... 642
31.3.8 Auxiliary Clock 9 Select Control Register (MC_CGM_AC9_SC)..... 643
31.3.9 Auxiliary Clock 9 Select Status Register (MC_CGM_AC9_SS)..... 644
31.3.10 PCS Switch Duration Register (MC_CGM_PCS_SDUR)..... 645
31.3.11 PCS Divider Change Register 1 (MC_CGM_PCS_DIVC1)..... 646

31.3.12 PCS Divider End Register 1 (MC_CGM_PCS_DIVE1)....646
31.3.13 PCS Divider Start Register 1 (MC_CGM_PCS_DIVS1)....647
31.3.14 PCS Divider Change Register 2 (MC_CGM_PCS_DIVC2)....648
31.3.15 PCS Divider End Register 2 (MC_CGM_PCS_DIVE2)....648
31.3.16 PCS Divider Start Register 2 (MC_CGM_PCS_DIVS2)....649
31.3.17 Divider Update Abort Register (MC_CGM_DIV_UPD_ABRTD)....650
31.3.18 System Clock Divider Ratio Change Register (MC_CGM_SC_DIV_RC)....651
31.3.19 Divider Update Type Register (MC_CGM_DIV_UPD_TYPE)....653
31.3.20 Divider Update Trigger Register (MC_CGM_DIV_UPD_TRIG)....654
31.3.21 Divider Update Status Register (MC_CGM_DIV_UPD_STAT)....655
31.3.22 System Clock Select Status Register (MC_CGM_SC_SS)....656
31.3.23 System Clock Divider 0 Configuration Register (MC_CGM_SC_DC0)....657
31.3.24 System Clock Divider 1 Configuration Register (MC_CGM_SC_DC1)....658
31.3.25 System Clock Divider 2 Configuration Register (MC_CGM_SC_DC2)....659
31.3.26 System Clock Divider 3 Configuration Register (MC_CGM_SC_DC3)....660
31.3.27 System Clock Divider 4 Configuration Register (MC_CGM_SC_DC4)....661
31.3.28 System Clock Divider 5 Configuration Register (MC_CGM_SC_DC5)....662
31.3.29 Auxiliary Clock 2 Select Control Register (MC_CGM_AC2_SC)....663
31.3.30 Auxiliary Clock 2 Select Status Register (MC_CGM_AC2_SS)....664
31.3.31 Auxiliary Clock 3 Select Status Register (MC_CGM_AC3_SS)....665
31.3.32 Auxiliary Clock 6 Select Control Register (MC_CGM_AC6_SC)....665
31.3.33 Auxiliary Clock 6 Select Status Register (MC_CGM_AC6_SS)....666
31.3.34 Aux Clock 6 Divider Configuration 0 for CLKOUT_0 (MC_CGM_AC6_DC0)....667
31.4 Functional Description....668
31.4.1 System Clock Generation....668
31.4.2 Auxiliary Clock Generation....674
31.4.3 Dividers Functional Description....676
31.4.4 Halt Abort....676

Chapter 32

Input Glitch Filter

32.1 Introduction....679
32.2 Block Diagram....679
32.3 Input Glitch Filter overview....680
32.3.1 Interaction with other peripherals....680
32.3.2 Mode of operation....680
32.4 Memory map....683
32.4.1 Module Configuration Register (IGF_MCR*n*)....684
32.4.2 Module Status Register (IGF_MSR*n*)....687
32.4.3 Prescaler Register (IGF_PRESR*n*)....688
32.4.4 Rising edge Threshold Register (IGF_RTHR*n*)....689
32.4.5 Falling edge Threshold Register (IGF_FTHR)....689
32.5 Functional description....689
32.5.1 Input double sampling....690
32.5.2 Filter types....690
32.5.3 Application....694
32.5.4 Programming guidelines....697
32.5.5 On-the-fly filter reconfiguration....697
32.6 Power saving mode....698
32.6.1 STOP mode....698

Chapter 33
HF - Interface

33.1 Chip specific HFIF information....701
33.1.1 Module Interconnect....701
33.2 Introduction....703
33.3 Block Diagram....703
33.4 Overview....705
33.4.1 Receiving Data....705
33.4.2 Transmitting Data....707

33.5 Functional Description....708
33.5.1 Clock Generation....708
33.5.2 Signal Level Timeout Detection....709
33.5.3 Pulse Detection....709
33.5.4 Interrupts....710
33.6 Memory Map and Registers....710
33.6.1 Clock Configuration Register (HFInterface_CCR)....711
33.6.2 Short Pulse Detector Configuration Register (HFInterface_SPDCR)....712
33.6.3 Long Pulse Detector Configuration Register (HFInterface_LPDCR)....713
33.6.4 Low-level Timeout Detector Configuration Register (HFInterface_LTDCR)....714
33.6.5 High-level Timeout Detector Configuration Register (HFInterface_HTDCR)....715
33.6.6 Interrupt Enable Register (HFInterface_IER)....716
33.6.7 Interrupt Status Register (HFInterface_ISR)....717

Chapter 34
Analog-to-Digital Converter (ADC)

34.1 Chip-specific ADC information....719
34.1.1 ADC configuration....719
34.1.2 Feature comparison....719
34.1.3 ADC register reset values....720
34.1.4 SARADC control registers....721
34.1.5 ADC channel mapping....725
34.1.6 ADC_0 valid control register bits....728
34.1.7 ADC_1 valid control register bits....738
34.1.8 BCTU Interface....746
34.1.9 DMA interface....746
34.1.10 ADC's stop-stop ack protocol....746
34.2 Introduction....746
34.3 Overview....746
34.3.1 Parts of the ADC....746

34.3.2 ADC clock signals ... 747
34.3.3 About ADC channels ... 748
34.4 Features ... 750
34.5 Memory Map/Register Definition ... 751
34.5.1 Main Configuration Register (ADC_MCR) ... 760
34.5.2 Main Status register (ADC_MSR) ... 763
34.5.3 Interrupt Status Register (ADC_ISR) ... 765
34.5.4 Channel Pending register 0 (ADC_CEOCFR0) ... 766
34.5.5 Channel Pending register 1 (ADC_CEOCFR1) ... 769
34.5.6 Channel Pending register 2 (ADC_CEOCFR2) ... 772
34.5.7 Interrupt Mask Register (ADC_IMR) ... 775
34.5.8 Channel Interrupt Mask Register 0 (ADC_CIMR0) ... 775
34.5.9 Channel Interrupt Mask Register 1 (ADC_CIMR1) ... 778
34.5.10 Channel Interrupt Mask Register 2 (ADC_CIMR2) ... 780
34.5.11 Watchdog Threshold Interrupt Status Register (ADC_WTISR) ... 783
34.5.12 Watchdog Threshold Interrupt Mask Register (ADC_WTIMR) ... 785
34.5.13 DMA Enable register (ADC_DMAE) ... 786
34.5.14 DMA Channel Select Register 0 (ADC_DMAR0) ... 787
34.5.15 DMA Channel Select Register 1 (ADC_DMAR1) ... 789
34.5.16 DMA Channel Select Register 2 (ADC_DMAR2) ... 792
34.5.17 Threshold Register (ADC_THRHLR*n*) ... 794
34.5.18 Presampling Control Register (ADC_PSCR) ... 795
34.5.19 Presampling register 0 (ADC_PSR0) ... 796
34.5.20 Presampling register 1 (ADC_PSR1) ... 798
34.5.21 Presampling register 2 (ADC_PSR2) ... 801
34.5.22 Conversion Timing Register 0 (ADC_CTR0) ... 803
34.5.23 Conversion Timing Register 1 (ADC_CTR1) ... 804
34.5.24 Conversion Timing Register 2 (ADC_CTR2) ... 804
34.5.25 Normal Conversion Mask Register 0 (ADC_NCMR0) ... 805

34.5.26 Normal Conversion Mask Register 1 (ADC_NCMR1)....807
34.5.27 Normal Conversion Mask Register 2 (ADC_NCMR2)....810
34.5.28 Injected Conversion Mask Register 0 (ADC_JCMR0)....812
34.5.29 Injected Conversion Mask Register 1 (ADC_JCMR1)....814
34.5.30 Injected Conversion Mask Register 2 (ADC_JCMR2)....817
34.5.31 Decode Signals Delay Register (ADC_DSDR)....820
34.5.32 Power Down Exit Delay Register (ADC_PDEDR)....820
34.5.33 Channel Data Register *n* (Precision Channels) (ADC_CDR*n*)....821
34.5.34 Channel Data Register *n* (Internal Channels) (ADC_CDR*n*)....823
34.5.35 Channel Data Register *n* (External Channels) (ADC_CDR*n*)....825
34.5.36 Threshold Register (ADC_THRHLR*n*)....826
34.5.37 Channel Watchdog Select Register 0 (ADC_CWSELR0)....826
34.5.38 Channel Watchdog Select Register 1 (ADC_CWSELR1)....827
34.5.39 Channel Watchdog Select Register 2 (ADC_CWSELR2)....828
34.5.40 Channel Watchdog Select Register 4 (ADC_CWSELR4)....829
34.5.41 Channel Watchdog Select Register 5 (ADC_CWSELR5)....830
34.5.42 Channel Watchdog Select Register 6 (ADC_CWSELR6)....831
34.5.43 Channel Watchdog Select Register 7 (ADC_CWSELR7)....832
34.5.44 Channel Watchdog Select Register 8 (ADC_CWSELR8)....833
34.5.45 Channel Watchdog Select Register 9 (ADC_CWSELR9)....834
34.5.46 Channel Watchdog Select Register 10 (ADC_CWSELR10)....835
34.5.47 Channel Watchdog Select Register 11 (ADC_CWSELR11)....836
34.5.48 Channel Watchdog Enable Register 0 (ADC_CWENR0)....837
34.5.49 Channel Watchdog Enable Register 1 (ADC_CWENR1)....840
34.5.50 Channel Watchdog Enable Register 2 (ADC_CWENR2)....842
34.5.51 Analog Watchdog Out of Range Register 0 (ADC_AWORR0)....845
34.5.52 Analog Watchdog Out of Range Register 1 (ADC_AWORR1)....848
34.5.53 Analog Watchdog Out of Range Register 2 (ADC_AWORR2)....850
34.5.54 Calibration, BIST Control and status Register (ADC_CALBISTREG)....853

34.5.55 Offset and Gain User Register (ADC_OFSGNUSR)..... 855
34.6 Functional description..... 856
34.6.1 Conversion..... 856
34.6.2 Crosstriggering Unit interface..... 862
34.6.3 ADC clock prescaler and sample time settings..... 864
34.6.4 Presampling..... 865
34.6.5 Programmable analog watchdog..... 866
34.6.6 DMA functionality..... 868
34.6.7 Interrupts..... 869
34.6.8 External Decode signals Delay..... 870
34.6.9 Power-Down mode..... 871
34.6.10 Auto-Clock-Off mode..... 871
34.6.11 Calibration..... 872
34.6.12 Conversion time..... 874
34.6.13 Conversion data processing..... 876
34.7 Clock frequency..... 877

Chapter 35
Analog Comparator (CMP)

35.1 Chip-specific CMP information..... 879
35.1.1 CMP configuration..... 879
35.1.2 CMP input output connections..... 880
35.1.3 CMP "VIN1" and "VIN2" references..... 881
35.1.4 Interaction with RTC API to cause wakeup..... 882
35.1.5 CMP Window control..... 883
35.2 Introduction..... 883
35.3 Features..... 884
35.3.1 CMP features..... 884
35.3.2 6-bit DAC key features..... 885
35.3.3 ANMUX key features..... 885

35.4 CMP, DAC, and ANMUX diagram....885
35.5 CMP block diagram....886
35.6 CMP pin descriptions....888
35.6.1 External pins....888
35.7 CMP functional modes....889
35.7.1 Disabled mode (# 1)....890
35.7.2 Continuous mode (#s 2A & 2B)....890
35.7.3 Sampled, Non-Filtered mode (#s 3A & 3B)....891
35.7.4 Sampled, Filtered mode (#s 4A & 4B)....893
35.7.5 Windowed mode (#s 5A & 5B)....895
35.7.6 Windowed/Resampled mode (# 6)....897
35.7.7 Windowed/Filtered mode (#7)....897
35.8 Memory map/register definitions....898
35.8.1 CMP Control Register 0 (CMP_C0)....899
35.8.2 CMP Control Register 1 (CMP_C1)....902
35.8.3 CMP Control Register 2 (CMP_C2)....904
35.9 CMP functional description....907
35.9.1 Initialization....907
35.9.2 Low-pass filter....907
35.10 Interrupts....910
35.11 DMA support....910
35.12 DAC functional description....910
35.12.1 Digital-to-analog converter block diagram....910
35.12.2 Voltage reference source select....911
35.13 DAC resets....911
35.14 DAC clocks....911
35.15 DAC interrupts....912
35.16 Trigger mode....912

Chapter 36

Body Cross triggering unit (BCTU)

36.1 Chip-specific BCTU information ... 915
36.1.1 BCTU configuration ... 915
36.1.2 Trigger sources ... 915
36.1.3 BCTU's stop-stop ack protocol ... 917
36.2 Introduction ... 917
36.3 BCTU overview ... 918
36.4 BCTU Interacting with other peripherals ... 919
36.5 BCTU Modes of operation ... 920
36.5.1 Configuration Mode ... 921
36.5.2 Normal Mode ... 921
36.5.3 Low Power Mode ... 921
36.6 Memory Map ... 922
36.6.1 Module Configuration Register (BCTU_MCR) ... 925
36.6.2 Module Status register (BCTU_MSR) ... 927
36.6.3 Trigger Configuration Register (BCTU_TRGCFG_*n*) ... 931
36.6.4 Write Protection register (BCTU_WRPROT) ... 933
36.6.5 Software Trigger register 1 (BCTU_SFTRGR1) ... 934
36.6.6 Software Trigger register 2 (BCTU_SFTRGR2) ... 937
36.6.7 ADC0 Result Data register (BCTU_ADC0DR) ... 940
36.6.8 ADC1 result data register (BCTU_ADC1DR) ... 941
36.6.9 LIST Size Status register (BCTU_LISTSTAR) ... 942
36.6.10 LIST Channel Address Register (BCTU_LISTCHR_*n*) ... 943
36.7 Functional description ... 944
36.7.1 Trigger Configuration ... 944
36.7.2 Trigger priority and multiplexing ... 945
36.7.3 Conversion LIST ... 947
36.7.4 ADC conversion results access ... 949
36.7.5 Trigger clearing mechanism ... 950

36.7.6 On-the-fly reconfiguration of BCTU registers 951
36.7.7 Multiple Parallel Conversions 952
36.8 Interrupts and DMA requests 953

Chapter 37
Configurable Enhanced Modular IO Subsystem (eMIOS)

37.1 Chip-specific eMIOS information 955
37.1.1 eMIOS configuration 955
37.1.2 Channel types 955
37.1.3 DMA channel configuration 957
37.1.4 Channel configuration 958
37.1.5 eMIOS disabling 958
37.1.6 Serialization 959
37.1.7 BCTU Interface 964
37.1.8 EMIOS register reset value 964
37.2 Introduction 964
37.2.1 Overview 964
37.2.2 Features 967
37.2.3 Modes of operation 967
37.3 Memory map / register definition 968
37.3.1 Memory map 968
37.4 Control Memory Map and Registers 969
37.4.1 eMIOS Module Configuration Register (eMIOS_MCR) 970
37.4.2 eMIOS Global FLAG Register (eMIOS_GFLAG) 972
37.4.3 eMIOS Output Update Disable register (eMIOS_OUDIS) 972
37.4.4 eMIOS Disable Channel register (eMIOS_UCDIS) 973
37.5 Unified Channels Memory Map and Registers 973
37.5.1 eMIOS UC A register n (eMIOS_A*n*) 979
37.5.2 eMIOS UC B register n (eMIOS_B*n*) 980
37.5.3 eMIOS CNT register n (eMIOS_CNT*n*) 981

37.5.4 eMIOS UC Control register n (eMIOS_C*n*)....982
37.5.5 eMIOS UC Status register n (eMIOS_S*n*)....988
37.5.6 eMIOS Alternate Address register n (eMIOS_ALTA*n*)....989
37.5.7 eMIOS UC Control 2 register n (eMIOS_C2_*n*)....990
37.6 Functional Description....991
37.6.1 Unified Channel (UC)....993
37.6.2 IP Bus Interface Unit (BIU)....1032
37.6.3 Global Clock Prescaler Submodule (GCP)....1033
37.7 Initialization/Application Information....1033
37.7.1 Considerations....1033
37.7.2 Application Information....1034

Chapter 38
Power Control Unit (MC_PCU)

38.1 Introduction....1039
38.1.1 Overview....1039
38.1.2 Features....1040

Chapter 39
Power Management Controller digital interface (PMCDIG)

39.1 Overview....1041
39.2 Features....1041
39.3 Memory Map and Registers....1042
39.3.1 RAM Domain Configuration Register (PMCDIG_RDCR)....1042
39.3.2 Supply Gauge Status Register (PMCDIG_SGSR)....1043
39.3.3 MISC Control Register (PMCDIG_MCR)....1047
39.4 Functional description....1048
39.4.1 Analog PMC interface....1049
39.4.2 Power-up Sequence....1049

Chapter 40
Mode Entry Module (MC_ME)

40.1 Introduction....1051

40.1.1 Overview....1051
40.1.2 Features....1053
40.1.3 Use Cases....1053
40.1.4 Modes of Operation....1060
40.2 External Signal Description....1062
40.3 Memory Map and Register Definition....1062
40.3.1 Global Status Register (MC_ME_GS)....1067
40.3.2 Mode Control Register (MC_ME_MCTL)....1070
40.3.3 Mode Enable Register (MC_ME_ME)....1072
40.3.4 Interrupt Status Register (MC_ME_IS)....1074
40.3.5 Interrupt Mask Register (MC_ME_IM)....1075
40.3.6 Invalid Mode Transition Status Register (MC_ME_IMTS)....1077
40.3.7 Debug Mode Transition Status Register (MC_ME_DMTS)....1079
40.3.8 RESET Mode Configuration Register (MC_ME_RESET_MC)....1084
40.3.9 SAFE Mode Configuration Register (MC_ME_SAFE_MC)....1086
40.3.10 DRUN Mode Configuration Register (MC_ME_DRUN_MC)....1088
40.3.11 RUN0 Mode Configuration Register (MC_ME_RUN0_MC)....1091
40.3.12 RUN1 Mode Configuration Register (MC_ME_RUN1_MC)....1093
40.3.13 RUN2 Mode Configuration Register (MC_ME_RUN2_MC)....1096
40.3.14 RUN3 Mode Configuration Register (MC_ME_RUN3_MC)....1098
40.3.15 STOP0 Mode Configuration Register (MC_ME_STOP_MC)....1100
40.3.16 STANDBY0 Mode Configuration Register (MC_ME_STANDBY_MC)....1103
40.3.17 Peripheral Status Register 0 (MC_ME_PS0)....1105
40.3.18 Peripheral Status Register 1 (MC_ME_PS1)....1108
40.3.19 Peripheral Status Register 2 (MC_ME_PS2)....1110
40.3.20 Peripheral Status Register 3 (MC_ME_PS3)....1112
40.3.21 Run Peripheral Configuration Register (MC_ME_RUN_PC*n*)....1114
40.3.22 Low-Power Peripheral Configuration Register (MC_ME_LP_PC*n*)....1115
40.3.23 BCTU Peripheral Control Register (MC_ME_PCTL0)....1116

40.3.24 eMIOS_0 Peripheral Control Register (MC_ME_PCTL1)..... 1117
40.3.25 eMIOS_1 Peripheral Control Register (MC_ME_PCTL2)..... 1118
40.3.26 SAI0 Peripheral Control Register (MC_ME_PCTL10).....1119
40.3.27 SAI1 Peripheral Control Register (MC_ME_PCTL11).....1120
40.3.28 SAI2 Peripheral Control Register (MC_ME_PCTL12).....1121
40.3.29 ENET Peripheral Control Register (MC_ME_PCTL15).....1122
40.3.30 CMP_0 Peripheral Control Register (MC_ME_PCTL20).....1123
40.3.31 CMP_1 Peripheral Control Register (MC_ME_PCTL21).....1124
40.3.32 CMP_2 Peripheral Control Register (MC_ME_PCTL22).....1125
40.3.33 ADC_0 Peripheral Control Register (MC_ME_PCTL24).....1126
40.3.34 ADC_1 Peripheral Control Register (MC_ME_PCTL25).....1127
40.3.35 FlexRay Peripheral Control Register (MC_ME_PCTL28)..... 1128
40.3.36 IIC_0 Peripheral Control Register (MC_ME_PCTL30).....1129
40.3.37 IIC_1 Peripheral Control Register (MC_ME_PCTL31).....1130
40.3.38 IIC_2 Peripheral Control Register (MC_ME_PCTL32).....1131
40.3.39 IIC_3 Peripheral Control Register (MC_ME_PCTL33).....1132
40.3.40 DSPI_0 Peripheral Control Register (MC_ME_PCTL40)..... 1133
40.3.41 DSPI_1 Peripheral Control Register (MC_ME_PCTL41)..... 1134
40.3.42 DSPI_2 Peripheral Control Register (MC_ME_PCTL42)..... 1135
40.3.43 DSPI_3 Peripheral Control Register (MC_ME_PCTL43)..... 1136
40.3.44 LIN_0 Peripheral Control Register (MC_ME_PCTL50)..... 1137
40.3.45 LIN_1 Peripheral Control Register (MC_ME_PCTL51)..... 1138
40.3.46 LIN_2 Peripheral Control Register (MC_ME_PCTL52)..... 1139
40.3.47 LIN_3 Peripheral Control Register (MC_ME_PCTL53)..... 1140
40.3.48 LIN_4 Peripheral Control Register (MC_ME_PCTL54)..... 1141
40.3.49 LIN_5 Peripheral Control Register (MC_ME_PCTL55)..... 1142
40.3.50 LIN_6 Peripheral Control Register (MC_ME_PCTL56)..... 1143
40.3.51 LIN_7 Peripheral Control Register (MC_ME_PCTL57)..... 1144
40.3.52 LIN 8 Peripheral Control Register (MC_ME_PCTL58)..... 1145

40.3.53 LIN_9 Peripheral Control Register (MC_ME_PCTL59)..... 1146
40.3.54 LIN_10 Peripheral Control Register (MC_ME_PCTL60)..... 1147
40.3.55 LIN_11 Peripheral Control Register (MC_ME_PCTL61)..... 1148
40.3.56 LIN 12 Peripheral Control Register (MC_ME_PCTL62)..... 1149
40.3.57 LIN_13 Peripheral Control Register (MC_ME_PCTL63)..... 1150
40.3.58 LIN 14 Peripheral Control Register (MC_ME_PCTL64)..... 1151
40.3.59 LIN_15 Peripheral Control Register (MC_ME_PCTL65)..... 1152
40.3.60 FLEXCAN_0 Peripheral Control Register (MC_ME_PCTL70).....1153
40.3.61 FLEXCAN_1 Peripheral Control Register (MC_ME_PCTL71).....1154
40.3.62 FLEXCAN_2 Peripheral Control Register (MC_ME_PCTL72).....1155
40.3.63 FLEXCAN_3 Peripheral Control Register (MC_ME_PCTL73).....1156
40.3.64 FLEXCAN_4 Peripheral Control Register (MC_ME_PCTL74).....1157
40.3.65 FLEXCAN_5 Peripheral Control Register (MC_ME_PCTL75).....1158
40.3.66 FLEXCAN_6 Peripheral Control Register (MC_ME_PCTL76).....1159
40.3.67 FLEXCAN_7 Peripheral Control Register (MC_ME_PCTL77).....1160
40.3.68 JDC Control Register (MC_ME_PCTL82)..... 1161
40.3.69 MEMU_0 Control Register (MC_ME_PCTL83)..... 1162
40.3.70 CRC Control Register (MC_ME_PCTL84)..... 1163
40.3.71 DMAMUX Control Register (MC_ME_PCTL90).....1164
40.3.72 PIT_RTI_0 Control Register (MC_ME_PCTL91)..... 1165
40.3.73 WKPU Control Register (MC_ME_PCTL93).....1166
40.3.74 SPI0 Control Register (MC_ME_PCTL96).....1167
40.3.75 SPI1 Control Register (MC_ME_PCTL97).....1168
40.3.76 SPI2 Control Register (MC_ME_PCTL98).....1169
40.3.77 SPI3 Control Register (MC_ME_PCTL99).....1170
40.3.78 RTC-API Control Register (MC_ME_PCTL102).....1171
40.3.79 MEMU_1 Control Register (MC_ME_PCTL105)..... 1172
40.3.80 Core Status Register (MC_ME_CS)..... 1174
40.3.81 Core1 Control Register (MC_ME_CCTL1)..... 1175

40.3.82 Core3 Control Register (MC_ME_CCTL3)..... 1176
40.3.83 CORE1 Address Register (MC_ME_CADDR1)..... 1178
40.3.84 CORE3 Address Register (MC_ME_CADDR3)..... 1179
40.4 Functional Description..... 1180
40.4.1 Mode Transition Request..... 1180
40.4.2 Modes Details..... 1181
40.4.3 Mode Transition Process..... 1187
40.4.4 Software Considerations for Preventing the Blocking of Mode Transitions..... 1200
40.4.5 Protection of Mode Configuration Registers..... 1202
40.4.6 Mode Transition Interrupts..... 1203
40.4.7 Peripheral Clock Gating..... 1206
40.4.8 Initialization/Application Example..... 1207

Chapter 41
General Programmable Registers (GPR)

41.1 Overview..... 1211
41.2 Memory map and register definition..... 1211
41.2.1 GPR Control Register (GPR_CTL)..... 1212
41.2.2 GPR Status Register (GPR_STAT)..... 1215
41.2.3 GPR Control Register1 (GPR_CTL1)..... 1216
41.2.4 GPR SLEEP Register (GPR_SLEEP)..... 1216

Chapter 42
Deserial Serial Peripheral Interface (DSPI)

42.1 Chip-specific DSPI information..... 1219
42.1.1 DSPI and SPI configuration..... 1219
42.1.2 DSPI configuration..... 1219
42.1.3 SPI configuration..... 1220
42.1.4 DSPI and SPI registers..... 1220
42.1.5 DSPI's stop-stop ack protocol..... 1222
42.2 Introduction..... 1222

42.2.1 Block Diagram ... 1222
42.2.2 Features ... 1223
42.2.3 Interface configurations ... 1226
42.2.4 Modes of Operation ... 1227
42.3 Module signal descriptions ... 1229
42.3.1 HT—Hardware Trigger ... 1229
42.3.2 PCS0/SS—Peripheral Chip Select/Slave Select ... 1230
42.3.3 PCS1–PCS3—Peripheral Chip Selects 1–3 ... 1230
42.3.4 PCS4/MTRIG—Peripheral Chip Select 4/Master Trigger ... 1230
42.3.5 PCS5/PCSS—Peripheral Chip Select 5/Peripheral Chip Select Strobe ... 1230
42.3.6 SCK—Serial Clock ... 1231
42.3.7 SIN—Serial Input ... 1231
42.3.8 SOUT—Serial Output ... 1231
42.4 Memory Map/Register Definition ... 1231
42.4.1 Module Configuration Register (DSPI_MCR) ... 1235
42.4.2 Transfer Count Register (DSPI_TCR) ... 1238
42.4.3 Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR*n*) ... 1239
42.4.4 Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR*n*_SLAVE) ... 1244
42.4.5 Status Register (DSPI_SR) ... 1246
42.4.6 DMA/Interrupt Request Select and Enable Register (DSPI_RSER) ... 1250
42.4.7 PUSH TX FIFO Register In Master Mode (DSPI_PUSHR) ... 1253
42.4.8 PUSH TX FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE) ... 1255
42.4.9 POP RX FIFO Register (DSPI_POPR) ... 1256
42.4.10 Transmit FIFO Registers (DSPI_TXFR*n*) ... 1256
42.4.11 Receive FIFO Registers (DSPI_RXFR*n*) ... 1257
42.4.12 DSI Configuration Register 0 (DSPI_DSICR0) ... 1258
42.4.13 DSI Serialization Data Register 0 (DSPI_SDR0) ... 1261
42.4.14 DSI Alternate Serialization Data Register 0 (DSPI_ASDR0) ... 1261
42.4.15 DSI Transmit Comparison Register 0 (DSPI_COMPR0) ... 1262

42.4.16 DSI Deserialization Data Register 0 (DSPI_DDR0)..... 1262
42.4.17 DSI Configuration Register 1 (DSPI_DSICR1)..... 1263
42.4.18 DSI Serialization Source Select Register 0 (DSPI_SSR0)..... 1265
42.4.19 DSI Parallel Input Select Register 0 (DSPI_PISR0)..... 1265
42.4.20 DSI Parallel Input Select Register 1 (DSPI_PISR1)..... 1266
42.4.21 DSI Parallel Input Select Register 2 (DSPI_PISR2)..... 1267
42.4.22 DSI Parallel Input Select Register 3 (DSPI_PISR3)..... 1268
42.4.23 DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)..... 1269
42.4.24 DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)..... 1269
42.4.25 DSI Serialization Data Register 1 (DSPI_SDR1)..... 1270
42.4.26 DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)..... 1271
42.4.27 DSI Transmit Comparison Register 1 (DSPI_COMPR1)..... 1271
42.4.28 DSI Deserialization Data Register 1 (DSPI_DDR1)..... 1272
42.4.29 DSI Serialization Source Select Register 1 (DSPI_SSR1)..... 1273
42.4.30 DSI Parallel Input Select Register 4 (DSPI_PISR4)..... 1273
42.4.31 DSI Parallel Input Select Register 5 (DSPI_PISR5)..... 1274
42.4.32 DSI Parallel Input Select Register 6 (DSPI_PISR6)..... 1275
42.4.33 DSI Parallel Input Select Register 7 (DSPI_PISR7)..... 1276
42.4.34 DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)..... 1277
42.4.35 DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)..... 1278
42.4.36 Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*)..... 1278
42.4.37 Status Register Extended (DSPI_SREX)..... 1280
42.4.38 Trigger Register (DSPI_TRIG)..... 1281
42.4.39 Time Slot Length Register (DSPI_TSL)..... 1282
42.4.40 Time Slot Configuration Register (DSPI_TS_CONF)..... 1283
42.5 Functional description..... 1285
42.5.1 Start and Stop of module transfers..... 1286
42.5.2 Serial Peripheral Interface (SPI) configuration..... 1287
42.5.3 Deserial Serial Interface (DSI) Configuration..... 1292

42.5.4 Combined Serial Interface (CSI) Configuration ... 1302
42.5.5 Module baud rate and clock delay generation ... 1304
42.5.6 Transfer formats ... 1307
42.5.7 Continuous Serial Communications Clock ... 1319
42.5.8 Slave Mode Operation Constraints ... 1321
42.5.9 Timed Serial Bus (TSB) ... 1321
42.5.10 Interleaved TSB (ITSB) Mode ... 1324
42.5.11 Parity Generation and Check ... 1333
42.5.12 Interrupts/DMA requests ... 1334
42.5.13 Power saving features ... 1339
42.6 Initialization/application information ... 1340
42.6.1 How to manage queues ... 1340
42.6.2 Switching Master and Slave mode ... 1341
42.6.3 Initializing Module in Master/Slave Modes ... 1341
42.6.4 Baud rate settings ... 1341
42.6.5 Delay settings ... 1342
42.6.6 Calculation of FIFO pointer addresses ... 1343

Chapter 43
Ethernet MAC (ENET)

43.1 Chip-specific ENET information ... 1347
43.1.1 ENET Configuration ... 1347
43.1.2 ENET Interrupt mapping ... 1347
43.1.3 ENET's stop-stop ack protocol ... 1348
43.1.4 Recieve Parser Match Array Table Depth ... 1348
43.2 Introduction ... 1349
43.3 Overview ... 1349
43.3.1 Features ... 1349
43.3.2 Block diagram ... 1352
43.4 External signal description ... 1353

43.5 Memory map/register definition..... 1355
43.5.1 Interrupt Event Register (ENET_EIR)..... 1362
43.5.2 Interrupt Mask Register (ENET_EIMR)..... 1365
43.5.3 Receive Descriptor Active Register - Ring 0 (ENET_RDAR)..... 1369
43.5.4 Transmit Descriptor Active Register - Ring 0 (ENET_TDAR)..... 1370
43.5.5 Ethernet Control Register (ENET_ECR)..... 1371
43.5.6 MII Management Frame Register (ENET_MMFR)..... 1373
43.5.7 MII Speed Control Register (ENET_MSCR)..... 1374
43.5.8 MIB Control Register (ENET_MIBC)..... 1376
43.5.9 Receive Control Register (ENET_RCR)..... 1377
43.5.10 Transmit Control Register (ENET_TCR)..... 1380
43.5.11 Physical Address Lower Register (ENET_PALR)..... 1382
43.5.12 Physical Address Upper Register (ENET_PAUR)..... 1382
43.5.13 Opcode/Pause Duration Register (ENET_OPD)..... 1383
43.5.14 Transmit Interrupt Coalescing Register (ENET_TXIC*n*)..... 1383
43.5.15 Receive Interrupt Coalescing Register (ENET_RXIC*n*)..... 1384
43.5.16 Descriptor Individual Upper Address Register (ENET_IAUR)..... 1385
43.5.17 Descriptor Individual Lower Address Register (ENET_IALR)..... 1386
43.5.18 Descriptor Group Upper Address Register (ENET_GAUR)..... 1386
43.5.19 Descriptor Group Lower Address Register (ENET_GALR)..... 1387
43.5.20 Transmit FIFO Watermark Register (ENET_TFWR)..... 1387
43.5.21 Receive Descriptor Ring 1 Start Register (ENET_RDSR1)..... 1388
43.5.22 Transmit Buffer Descriptor Ring 1 Start Register (ENET_TDSR1)..... 1389
43.5.23 Maximum Receive Buffer Size Register - Ring 1 (ENET_MRBR1)..... 1390
43.5.24 Receive Descriptor Ring 2 Start Register (ENET_RDSR2)..... 1391
43.5.25 Transmit Buffer Descriptor Ring 2 Start Register (ENET_TDSR2)..... 1391
43.5.26 Maximum Receive Buffer Size Register - Ring 2 (ENET_MRBR2)..... 1392
43.5.27 Receive Descriptor Ring 0 Start Register (ENET_RDSR)..... 1393
43.5.28 Transmit Buffer Descriptor Ring 0 Start Register (ENET_TDSR)..... 1394

43.5.29 Maximum Receive Buffer Size Register - Ring 0 (ENET_MRBR)..... 1394
43.5.30 Receive FIFO Section Full Threshold (ENET_RSFL)..... 1395
43.5.31 Receive FIFO Section Empty Threshold (ENET_RSEM)..... 1396
43.5.32 Receive FIFO Almost Empty Threshold (ENET_RAEM)..... 1396
43.5.33 Receive FIFO Almost Full Threshold (ENET_RAFL)..... 1397
43.5.34 Transmit FIFO Section Empty Threshold (ENET_TSEM)..... 1397
43.5.35 Transmit FIFO Almost Empty Threshold (ENET_TAEM)..... 1398
43.5.36 Transmit FIFO Almost Full Threshold (ENET_TAFL)..... 1398
43.5.37 Transmit Inter-Packet Gap (ENET_TIPG)..... 1399
43.5.38 Frame Truncation Length (ENET_FTRL)..... 1399
43.5.39 Transmit Accelerator Function Configuration (ENET_TACC)..... 1400
43.5.40 Receive Accelerator Function Configuration (ENET_RACC)..... 1401
43.5.41 Receive Classification Match Register for Class n (ENET_RCMR*n*)..... 1403
43.5.42 DMA Class Based Configuration (ENET_DMA*n*CFG)..... 1404
43.5.43 Receive Descriptor Active Register - Ring 1 (ENET_RDAR1)..... 1406
43.5.44 Transmit Descriptor Active Register - Ring 1 (ENET_TDAR1)..... 1406
43.5.45 Receive Descriptor Active Register - Ring 2 (ENET_RDAR2)..... 1408
43.5.46 Transmit Descriptor Active Register - Ring 2 (ENET_TDAR2)..... 1409
43.5.47 QOS Scheme (ENET_QOS)..... 1409
43.5.48 Reserved Statistic Register (ENET_RMON_T_DROP)..... 1411
43.5.49 Tx Packet Count Statistic Register (ENET_RMON_T_PACKETS)..... 1411
43.5.50 Tx Broadcast Packets Statistic Register (ENET_RMON_T_BC_PKT)..... 1412
43.5.51 Tx Multicast Packets Statistic Register (ENET_RMON_T_MC_PKT)..... 1412
43.5.52 Tx Packets with CRC/Align Error Statistic Register (ENET_RMON_T_CRC_ALIGN)..... 1413
43.5.53 Tx Packets Less Than Bytes and Good CRC Statistic Register (ENET_RMON_T_UNDERSIZE)..... 1413
43.5.54 Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (ENET_RMON_T_OVERSIZE)..... 1414
43.5.55 Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_T_FRAG)..... 1414
43.5.56 Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (ENET_RMON_T_JAB)..... 1415
43.5.57 Tx Collision Count Statistic Register (ENET_RMON_T_COL)..... 1415

43.5.58 Tx 64-Byte Packets Statistic Register (ENET_RMON_T_P64)..... 1416
43.5.59 Tx 65- to 127-byte Packets Statistic Register (ENET_RMON_T_P65TO127)..... 1416
43.5.60 Tx 128- to 255-byte Packets Statistic Register (ENET_RMON_T_P128TO255)..... 1417
43.5.61 Tx 256- to 511-byte Packets Statistic Register (ENET_RMON_T_P256TO511)..... 1417
43.5.62 Tx 512- to 1023-byte Packets Statistic Register (ENET_RMON_T_P512TO1023)..... 1418
43.5.63 Tx 1024- to 2047-byte Packets Statistic Register (ENET_RMON_T_P1024TO2047)..... 1418
43.5.64 Tx Packets Greater Than 2048 Bytes Statistic Register (ENET_RMON_T_P_GTE2048)..... 1419
43.5.65 Tx Octets Statistic Register (ENET_RMON_T_OCTETS)..... 1419
43.5.66 Reserved Statistic Register (ENET_IEEE_T_DROP)..... 1419
43.5.67 Frames Transmitted OK Statistic Register (ENET_IEEE_T_FRAME_OK)..... 1420
43.5.68 Frames Transmitted with Single Collision Statistic Register (ENET_IEEE_T_1COL)..... 1420
43.5.69 Frames Transmitted with Multiple Collisions Statistic Register (ENET_IEEE_T_MCOL)..... 1421
43.5.70 Frames Transmitted after Deferral Delay Statistic Register (ENET_IEEE_T_DEF)..... 1421
43.5.71 Frames Transmitted with Late Collision Statistic Register (ENET_IEEE_T_LCOL)..... 1422
43.5.72 Frames Transmitted with Excessive Collisions Statistic Register (ENET_IEEE_T_EXCOL)..... 1422
43.5.73 Frames Transmitted with Tx FIFO Underrun Statistic Register (ENET_IEEE_T_MACERR)..... 1423
43.5.74 Frames Transmitted with Carrier Sense Error Statistic Register (ENET_IEEE_T_CSERR)..... 1423
43.5.75 Reserved Statistic Register (ENET_IEEE_T_SQE)..... 1424
43.5.76 Flow Control Pause Frames Transmitted Statistic Register (ENET_IEEE_T_FDXFC)..... 1424
43.5.77 Octet Count for Frames Transmitted w/o Error Statistic Register (ENET_IEEE_T_OCTETS_OK)..... 1425
43.5.78 Rx Packet Count Statistic Register (ENET_RMON_R_PACKETS)..... 1425
43.5.79 Rx Broadcast Packets Statistic Register (ENET_RMON_R_BC_PKT)..... 1426
43.5.80 Rx Multicast Packets Statistic Register (ENET_RMON_R_MC_PKT)..... 1426
43.5.81 Rx Packets with CRC/Align Error Statistic Register (ENET_RMON_R_CRC_ALIGN)..... 1427
43.5.82 Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (ENET_RMON_R_UNDERSIZE)..... 1427
43.5.83 Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (ENET_RMON_R_OVERSIZE). 1428
43.5.84 Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_R_FRAG)..... 1428
43.5.85 Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (ENET_RMON_R_JAB)..... 1429

43.5.86 Reserved Statistic Register (ENET_RMON_R_RESVD_0)..... 1429
43.5.87 Rx 64-Byte Packets Statistic Register (ENET_RMON_R_P64)..... 1429
43.5.88 Rx 65- to 127-Byte Packets Statistic Register (ENET_RMON_R_P65TO127)..... 1430
43.5.89 Rx 128- to 255-Byte Packets Statistic Register (ENET_RMON_R_P128TO255)..... 1430
43.5.90 Rx 256- to 511-Byte Packets Statistic Register (ENET_RMON_R_P256TO511)..... 1431
43.5.91 Rx 512- to 1023-Byte Packets Statistic Register (ENET_RMON_R_P512TO1023)..... 1431
43.5.92 Rx 1024- to 2047-Byte Packets Statistic Register (ENET_RMON_R_P1024TO2047)..... 1432
43.5.93 Rx Packets Greater than 2048 Bytes Statistic Register (ENET_RMON_R_P_GTE2048)..... 1432
43.5.94 Rx Octets Statistic Register (ENET_RMON_R_OCTETS)..... 1433
43.5.95 Frames not Counted Correctly Statistic Register (ENET_IEEE_R_DROP)..... 1433
43.5.96 Frames Received OK Statistic Register (ENET_IEEE_R_FRAME_OK)..... 1434
43.5.97 Frames Received with CRC Error Statistic Register (ENET_IEEE_R_CRC)..... 1434
43.5.98 Frames Received with Alignment Error Statistic Register (ENET_IEEE_R_ALIGN)..... 1435
43.5.99 Receive FIFO Overflow Count Statistic Register (ENET_IEEE_R_MACERR)..... 1435
43.5.100 Flow Control Pause Frames Received Statistic Register (ENET_IEEE_R_FDXFC)..... 1436
43.5.101 Octet Count for Frames Received without Error Statistic Register (ENET_IEEE_R_OCTETS_OK)..... 1436
43.5.102 Adjustable Timer Control Register (ENET_ATCR)..... 1436
43.5.103 Timer Value Register (ENET_ATVR)..... 1438
43.5.104 Timer Offset Register (ENET_ATOFF)..... 1439
43.5.105 Timer Period Register (ENET_ATPER)..... 1439
43.5.106 Timer Correction Register (ENET_ATCOR)..... 1440
43.5.107 Time-Stamping Clock Period Register (ENET_ATINC)..... 1440
43.5.108 Timestamp of Last Transmitted Frame (ENET_ATSTMP)..... 1441
43.5.109 Pattern Match Data Register (ENET_MDATA)..... 1441
43.5.110 Match Entry Mask Register (ENET_MMASK)..... 1442
43.5.111 Match Entry Rules Configuration Register (ENET_MCONFIG)..... 1442
43.5.112 Match Entry Read/Write Command Register (ENET_MENTRYRW)..... 1443
43.5.113 Receive Parser Control Register (ENET_RXPCTL)..... 1444
43.5.114 Maximum Frame Offset (ENET_MAXFRMOFF)..... 1446

43.5.115 Receive Parser Status (ENET_RXPARST)..... 1447
43.5.116 Parser Discard Count (ENET_PARSDSCD)..... 1448
43.5.117 Parser Accept Count 0 (ENET_PRSACPT0)..... 1449
43.5.118 Parser Reject Count 0 (ENET_PRSRJCT0)..... 1449
43.5.119 Parser Accept Count 1 (ENET_PRSACPT1)..... 1450
43.5.120 Parser Reject Count 1 (ENET_PRSRJCT1)..... 1450
43.5.121 Parser Accept Count 2 (ENET_PRSACPT2)..... 1451
43.5.122 Parser Reject Count 2 (ENET_PRSRJCT2)..... 1451
43.5.123 Timer Global Status Register (ENET_TGSR)..... 1452
43.5.124 Timer Control Status Register (ENET_TCSR*n*)..... 1453
43.5.125 Timer Compare Capture Register (ENET_TCCR*n*)..... 1454
43.6 Functional description..... 1455
43.6.1 Ethernet MAC frame formats..... 1455
43.6.2 IP and higher layers frame format..... 1458
43.6.3 IEEE 1588 message formats..... 1462
43.6.4 MAC receive..... 1466
43.6.5 MAC transmit..... 1475
43.6.6 Full-duplex flow control operation..... 1481
43.6.7 Magic packet detection..... 1483
43.6.8 IP accelerator functions..... 1484
43.6.9 Resets and stop controls..... 1488
43.6.10 IEEE 1588 functions..... 1491
43.6.11 FIFO thresholds..... 1495
43.6.12 Loopback options..... 1498
43.6.13 Legacy buffer descriptors..... 1499
43.6.14 Enhanced buffer descriptors..... 1500
43.6.15 Client FIFO application interface..... 1507
43.6.16 FIFO protection..... 1510
43.6.17 Reference clock..... 1512

43.6.18 PHY management interface ... 1512
43.6.19 Ethernet interfaces ... 1515
43.6.20 AVB configuration ... 1519
43.6.21 Interrupt coalescence ... 1520

Chapter 44
FlexCAN

44.1 Chip-specific FlexCAN information ... 1523
44.1.1 FlexCAN configuration ... 1523
44.1.2 Requirements for entering FlexCAN modes: Freeze, Module Disable, Stop ... 1524
44.1.3 External time tick ... 1526
44.1.4 FlexCAN's stop-stop ack protocol ... 1527
44.2 Introduction ... 1527
44.2.1 Overview ... 1528
44.2.2 FlexCAN module features ... 1529
44.2.3 Modes of operation ... 1531
44.3 FlexCAN signal descriptions ... 1533
44.3.1 CAN Rx ... 1533
44.3.2 CAN Tx ... 1533
44.4 Memory map/register definition ... 1533
44.4.1 FlexCAN memory mapping ... 1533
44.4.2 Module Configuration Register (CAN_MCR) ... 1542
44.4.3 Control 1 register (CAN_CTRL1) ... 1547
44.4.4 Free Running Timer (CAN_TIMER) ... 1551
44.4.5 Rx Mailboxes Global Mask Register (CAN_RXMGMASK) ... 1552
44.4.6 Rx 14 Mask register (CAN_RX14MASK) ... 1553
44.4.7 Rx 15 Mask register (CAN_RX15MASK) ... 1554
44.4.8 Error Counter (CAN_ECR) ... 1555
44.4.9 Error and Status 1 register (CAN_ESR1) ... 1557
44.4.10 Interrupt Masks 2 register (CAN_IMASK2) ... 1564

44.4.11 Interrupt Masks 1 register (CAN_IMASK1)..... 1564
44.4.12 Interrupt Flags 2 register (CAN_IFLAG2)..... 1565
44.4.13 Interrupt Flags 1 register (CAN_IFLAG1)..... 1566
44.4.14 Control 2 register (CAN_CTRL2)..... 1569
44.4.15 Error and Status 2 register (CAN_ESR2)..... 1572
44.4.16 CRC Register (CAN_CRCR)..... 1574
44.4.17 Rx FIFO Global Mask register (CAN_RXFGMASK)..... 1574
44.4.18 Rx FIFO Information Register (CAN_RXFIR)..... 1575
44.4.19 CAN Bit Timing Register (CAN_CBT)..... 1576
44.4.20 Interrupt Masks 3 Register (CAN_IMASK3)..... 1578
44.4.21 Interrupt Flags 3 Register (CAN_IFLAG3)..... 1579
44.4.22 Rx Individual Mask Registers (CAN_RXIMR*n*)..... 1579
44.4.23 Pretended Networking Control 1 Register (CAN_CTRL1_PN)..... 1581
44.4.24 Pretended Networking Control 2 Register (CAN_CTRL2_PN)..... 1582
44.4.25 Pretended Networking Wake Up Match Register (CAN_WU_MTC)..... 1583
44.4.26 Pretended Networking ID Filter 1 Register (CAN_FLT_ID1)..... 1585
44.4.27 Pretended Networking DLC Filter Register (CAN_FLT_DLC)..... 1586
44.4.28 Pretended Networking Payload Low Filter 1 Register (CAN_PL1_LO)..... 1587
44.4.29 Pretended Networking Payload High Filter 1 Register (CAN_PL1_HI)..... 1587
44.4.30 Pretended Networking ID Filter 2 Register / ID Mask Register (CAN_FLT_ID2_IDMASK)..... 1588
44.4.31 Pretended Networking Payload Low Filter 2 Register / Payload Low Mask Register (CAN_PL2_PLMASK_LO)..... 1589
44.4.32 Pretended Networking Payload High Filter 2 low order bits / Payload High Mask Register (CAN_PL2_PLMASK_HI)..... 1590
44.4.33 Wake Up Message Buffer Register for C/S (CAN_WMB*n*_CS)..... 1590
44.4.34 Wake Up Message Buffer Register for ID (CAN_WMB*n*_ID)..... 1591
44.4.35 Wake Up Message Buffer Register for Data 0-3 (CAN_WMB*n*_D03)..... 1592
44.4.36 Wake Up Message Buffer Register Data 4-7 (CAN_WMB*n*_D47)..... 1593
44.4.37 CAN FD Control Register (CAN_FDCTRL)..... 1593

44.4.38 CAN FD Bit Timing Register (CAN_FDCBT)..... 1597
44.4.39 CAN FD CRC Register (CAN_FDCRC)..... 1599
44.4.40 Message buffer structure..... 1601
44.4.41 FlexCAN Memory Partition for CAN FD..... 1607
44.4.42 FlexCAN message buffer memory map..... 1609
44.4.43 Rx FIFO structure..... 1615
44.5 Functional description..... 1617
44.5.1 Transmit process..... 1618
44.5.2 Arbitration process..... 1619
44.5.3 Receive process..... 1623
44.5.4 Matching process..... 1625
44.5.5 Receive Process under Pretended Networking Mode..... 1629
44.5.6 Move process..... 1634
44.5.7 Data coherence..... 1636
44.5.8 Rx FIFO..... 1639
44.5.9 CAN protocol related features..... 1642
44.5.10 Clock domains and restrictions..... 1662
44.5.11 Modes of operation details..... 1667
44.5.12 Interrupts..... 1671
44.5.13 Bus interface..... 1672
44.6 Initialization/application information..... 1673
44.6.1 FlexCAN initialization sequence..... 1673

Chapter 45
Inter-Integrated Circuit (I2C)

45.1 Chip-specific I2C information..... 1677
45.1.1 I2C configurations..... 1677
45.2 Overview..... 1677
45.3 Introduction to I2C..... 1677
45.3.1 Definition: I2C module..... 1678

45.3.2 Advantages of the I2C bus ... 1678
45.3.3 Module block diagram ... 1678
45.3.4 Features ... 1679
45.3.5 Modes of operation ... 1680
45.3.6 Definition: I2C conditions ... 1681
45.4 External signal descriptions ... 1682
45.4.1 Signal overview ... 1682
45.4.2 Detailed external signal descriptions ... 1682
45.5 Memory map and register definition ... 1682
45.5.1 Register accessibility ... 1683
45.5.2 Register figure conventions ... 1683
45.5.3 I2C Bus Address Register (I2C_IBAD) ... 1684
45.5.4 I2C Bus Frequency Divider Register (I2C_IBFD) ... 1685
45.5.5 I2C Bus Control Register (I2C_IBCR) ... 1685
45.5.6 I2C Bus Status Register (I2C_IBSR) ... 1687
45.5.7 I2C Bus Data I/O Register (I2C_IBDR) ... 1688
45.5.8 I2C Bus Interrupt Config Register (I2C_IBIC) ... 1689
45.5.9 I2C Bus Debug Register (I2C_IBDBG) ... 1690
45.6 Functional description ... 1691
45.6.1 Notes about module operation ... 1691
45.6.2 Transactions ... 1691
45.6.3 Arbitration procedure ... 1695
45.6.4 Clock behavior ... 1696
45.6.5 Interrupts ... 1703
45.6.6 STOP mode ... 1704
45.6.7 DEBUG mode ... 1705
45.6.8 DMA interface ... 1707
45.7 Initialization/application information ... 1708
45.7.1 Recommended interrupt service flow ... 1708

45.7.2 General programming guidelines (for both master and slave mode)....1709
45.7.3 Programming guidelines specific to master mode....1711
45.7.4 Programming guidelines specific to slave mode....1715
45.7.5 DMA application information....1715

Chapter 46
LINFlexD

46.1 Chip-specific LINFlexD information....1723
46.1.1 LINFlexD configurations....1723
46.1.2 LinflexD's stop-stop ack protocol....1724
46.2 Introduction....1724
46.2.1 Glossary and acronyms....1724
46.2.2 References....1725
46.3 Main features....1727
46.3.1 LIN mode features....1727
46.3.2 UART mode features....1728
46.4 Functional description....1729
46.4.1 LIN protocol....1729
46.4.2 LINFlexD features....1731
46.4.3 Timer....1751
46.4.4 UART mode....1751
46.4.5 DMA interface....1757
46.5 Memory map and register description....1775
46.5.1 LIN Control Register 1 (LINFlexD_LINCR1)....1777
46.5.2 LIN Interrupt enable register (LINFlexD_LINIER)....1780
46.5.3 LIN Status Register (LINFlexD_LINSR)....1782
46.5.4 LIN Error Status Register (LINFlexD_LINESR)....1786
46.5.5 UART Mode Control Register (LINFlexD_UARTCR)....1787
46.5.6 UART Mode Status Register (LINFlexD_UARTSR)....1793
46.5.7 LIN Time-Out Control Status Register (LINFlexD_LINTCSR)....1795

46.5.8 LIN Output Compare Register (LINFlexD_LINOCR)....1797
46.5.9 LIN Time-Out Control Register (LINFlexD_LINTOCR)....1798
46.5.10 LIN Fractional Baud Rate Register (LINFlexD_LINFBRR)....1799
46.5.11 LIN Integer Baud Rate Register (LINFlexD_LINIBRR)....1800
46.5.12 LIN Checksum Field Register (LINFlexD_LINCFR)....1800
46.5.13 LIN Control Register 2 (LINFlexD_LINCR2)....1801
46.5.14 Buffer Identifier Register (LINFlexD_BIDR)....1803
46.5.15 Buffer Data Register Least Significant (LINFlexD_BDRL)....1804
46.5.16 Buffer Data Register Most Significant (LINFlexD_BDRM)....1805
46.5.17 Identifier Filter Enable Register (LINFlexD_IFER)....1805
46.5.18 Identifier Filter Match Index (LINFlexD_IFMI)....1806
46.5.19 Identifier Filter Mode Register (LINFlexD_IFMR)....1806
46.5.20 Identifier Filter Control Register (LINFlexD_IFCR*n*)....1807
46.5.21 Global Control Register (LINFlexD_GCR)....1808
46.5.22 UART Preset Timeout Register (LINFlexD_UARTPTO)....1810
46.5.23 UART Current Timeout Register (LINFlexD_UARTCTO)....1811
46.5.24 DMA Tx Enable Register (LINFlexD_DMATXE)....1812
46.5.25 DMA Rx Enable Register (LINFlexD_DMARXE)....1812
46.6 Programming considerations....1813
46.6.1 Master node....1813
46.6.2 Slave node....1815
46.6.3 Timeout....1820
46.6.4 UART mode....1821
46.6.5 Interrupts....1822
46.6.6 LINFlexD Clock Tolerance....1823

Chapter 47
FlexRay

47.1 Chip-specific FlexRay information....1825
47.1.1 FlexRay configuration....1825

47.1.2 Stop Watch interface....1825
47.2 Introduction....1825
47.2.1 Reference....1825
47.2.2 Glossary....1826
47.2.3 Overview....1827
47.2.4 Features....1829
47.2.5 Modes of Operation....1831
47.3 External Signal Description....1833
47.3.1 Detailed Signal Descriptions....1833
47.4 Controller Host Interface Clocking....1835
47.5 Protocol Engine Clocking....1835
47.5.1 Oscillator Clocking....1835
47.5.2 PLL Clocking....1835
47.6 Register Descriptions....1836
47.6.1 Register Reset....1836
47.6.2 Register Write Access....1837
47.7 Memory map and register definition....1838
47.7.1 Module Version Register (FR_MVR)....1875
47.7.2 Module Configuration Register (FR_MCR)....1875
47.7.3 System Memory Base Address High Register (FR_SYMBADHR)....1878
47.7.4 System Memory Base Address Low Register (FR_SYMBADLR)....1878
47.7.5 Strobe Signal Control Register (FR_STBSCR)....1879
47.7.6 Message Buffer Data Size Register (FR_MBDSR)....1881
47.7.7 Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)....1881
47.7.8 PE DRAM Access Register (FR_PEDRAR)....1882
47.7.9 PE DRAM Data Register (FR_PEDRDR)....1883
47.7.10 Protocol Operation Control Register (FR_POCR)....1884
47.7.11 Global Interrupt Flag and Enable Register (FR_GIFER)....1885
47.7.12 Protocol Interrupt Flag Register 0 (FR_PIFR0)....1888

47.7.13 Protocol Interrupt Flag Register 1 (FR_PIFR1)....1890
47.7.14 Protocol Interrupt Enable Register 0 (FR_PIER0)....1892
47.7.15 Protocol Interrupt Enable Register 1 (FR_PIER1)....1894
47.7.16 CHI Error Flag Register (FR_CHIERFR)....1895
47.7.17 Message Buffer Interrupt Vector Register (FR_MBIVEC)....1898
47.7.18 Channel A Status Error Counter Register (FR_CASERCR)....1899
47.7.19 Channel B Status Error Counter Register (FR_CBSERCR)....1899
47.7.20 Protocol Status Register 0 (FR_PSR0)....1900
47.7.21 Protocol Status Register 1 (FR_PSR1)....1902
47.7.22 Protocol Status Register 2 (FR_PSR2)....1903
47.7.23 Protocol Status Register 3 (FR_PSR3)....1905
47.7.24 Macrotick Counter Register (FR_MTCTR)....1907
47.7.25 Cycle Counter Register (FR_CYCTR)....1908
47.7.26 Slot Counter Channel A Register (FR_SLTCTAR)....1908
47.7.27 Slot Counter Channel B Register (FR_SLTCTBR)....1909
47.7.28 Rate Correction Value Register (FR_RTCORVR)....1909
47.7.29 Offset Correction Value Register (FR_OFCORVR)....1910
47.7.30 Combined Interrupt Flag Register (FR_CIFR)....1911
47.7.31 System Memory Access Time-Out Register (FR_SYMATOR)....1912
47.7.32 Sync Frame Counter Register (FR_SFCNTR)....1913
47.7.33 Sync Frame Table Offset Register (FR_SFTOR)....1913
47.7.34 Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)....1914
47.7.35 Sync Frame ID Rejection Filter Register (FR_SFIDRFR)....1916
47.7.36 Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)....1916
47.7.37 Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)....1917
47.7.38 Network Management Vector Register (FR_NMVR*n*)....1917
47.7.39 Network Management Vector Length Register (FR_NMVLR)....1918
47.7.40 Timer Configuration and Control Register (FR_TICCR)....1918
47.7.41 Timer 1 Cycle Set Register (FR_TI1CYSR)....1920

47.7.42 Timer 1 Macrotick Offset Register (FR_TI1MTOR)..... 1921
47.7.43 Timer 2 Configuration Register 0 (Absolute Timer Configuration) (FR_TI2CR0_ABS)..... 1921
47.7.44 Timer 2 Configuration Register 0 (Relative Timer Configuration) (FR_TI2CR0_REL).....1922
47.7.45 Timer 2 Configuration Register 1 (Absolute Timer Configuration) (FR_TI2CR1_ABS)..... 1922
47.7.46 Timer 2 Configuration Register 1 (Relative Timer Configuration) (FR_TI2CR1_REL).....1923
47.7.47 Slot Status Selection Register (FR_SSSR)..... 1924
47.7.48 Slot Status Counter Condition Register (FR_SSCCR)..... 1925
47.7.49 Slot Status Register (FR_SSR*n*)..... 1927
47.7.50 Slot Status Counter Register (FR_SSCR*n*)..... 1929
47.7.51 MTS A Configuration Register (FR_MTSACFR).....1929
47.7.52 MTS B Configuration Register (FR_MTSBCFR)..... 1930
47.7.53 Receive Shadow Buffer Index Register (FR_RSBIR)..... 1931
47.7.54 Receive FIFO Watermark and Selection Register (FR_RFWMSR)..... 1932
47.7.55 Receive FIFO Start Index Register (FR_RFSIR)..... 1933
47.7.56 Receive FIFO Depth and Size Register (FR_RFDSR)..... 1933
47.7.57 Receive FIFO A Read Index Register (FR_RFARIR).....1934
47.7.58 Receive FIFO B Read Index Register (FR_RFBRIR)..... 1934
47.7.59 Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR).....1935
47.7.60 Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR).....1935
47.7.61 Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)..... 1936
47.7.62 Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)..... 1936
47.7.63 Receive FIFO Range Filter Configuration Register (FR_RFRFCFR).....1937
47.7.64 Receive FIFO Range Filter Control Register (FR_RFRFCTR).....1938
47.7.65 Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)..... 1939
47.7.66 Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR).....1940
47.7.67 Protocol Configuration Register 0 (FR_PCR0)..... 1940
47.7.68 Protocol Configuration Register 1 (FR_PCR1)..... 1943
47.7.69 Protocol Configuration Register 2 (FR_PCR2)..... 1943
47.7.70 Protocol Configuration Register 3 (FR_PCR3)..... 1944

47.7.71 Protocol Configuration Register 4 (FR_PCR4)..... 1944
47.7.72 Protocol Configuration Register 5 (FR_PCR5)..... 1945
47.7.73 Protocol Configuration Register 6 (FR_PCR6)..... 1945
47.7.74 Protocol Configuration Register 7 (FR_PCR7)..... 1946
47.7.75 Protocol Configuration Register 8 (FR_PCR8)..... 1946
47.7.76 Protocol Configuration Register 9 (FR_PCR9)..... 1947
47.7.77 Protocol Configuration Register 10 (FR_PCR10)..... 1948
47.7.78 Protocol Configuration Register 11 (FR_PCR11)..... 1948
47.7.79 Protocol Configuration Register 12 (FR_PCR12)..... 1949
47.7.80 Protocol Configuration Register 13 (FR_PCR13)..... 1950
47.7.81 Protocol Configuration Register 14 (FR_PCR14)..... 1950
47.7.82 Protocol Configuration Register 15 (FR_PCR15)..... 1951
47.7.83 Protocol Configuration Register 16 (FR_PCR16)..... 1951
47.7.84 Protocol Configuration Register 17 (FR_PCR17)..... 1952
47.7.85 Protocol Configuration Register 18 (FR_PCR18)..... 1952
47.7.86 Protocol Configuration Register 19 (FR_PCR19)..... 1953
47.7.87 Protocol Configuration Register 20 (FR_PCR20)..... 1953
47.7.88 Protocol Configuration Register 21 (FR_PCR21)..... 1954
47.7.89 Protocol Configuration Register 22 (FR_PCR22)..... 1954
47.7.90 Protocol Configuration Register 23 (FR_PCR23)..... 1955
47.7.91 Protocol Configuration Register 24 (FR_PCR24)..... 1955
47.7.92 Protocol Configuration Register 25 (FR_PCR25)..... 1956
47.7.93 Protocol Configuration Register 26 (FR_PCR26)..... 1956
47.7.94 Protocol Configuration Register 27 (FR_PCR27)..... 1957
47.7.95 Protocol Configuration Register 28 (FR_PCR28)..... 1958
47.7.96 Protocol Configuration Register 29 (FR_PCR29)..... 1958
47.7.97 Protocol Configuration Register 30 (FR_PCR30)..... 1959
47.7.98 StopWatch Count High Register (FR_STPWHR)..... 1959
47.7.99 Stop Watch Count Low Register (FR_STPWLR)..... 1960

47.7.100 Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)..... 1960
47.7.101 Receive FIFO Start Data Offset Register (FR_RFSDOR).....1961
47.7.102 Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)..... 1962
47.7.103 Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)..... 1962
47.7.104 Receive FIFO Periodic Timer Register (FR_RFPTR).....1963
47.7.105 Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR).....1963
47.7.106 ECC Error Interrupt Flag and Enable Register (FR_EEIFER)..... 1965
47.7.107 ECC Error Report and Injection Control Register (FR_EERICR)..... 1967
47.7.108 ECC Error Report Address Register (FR_EERAR)..... 1968
47.7.109 ECC Error Report Data Register (FR_EERDR)..... 1969
47.7.110 ECC Error Report Code Register (FR_EERCR)..... 1970
47.7.111 ECC Error Injection Address Register (FR_EEIAR)..... 1971
47.7.112 ECC Error Injection Data Register (FR_EEIDR)..... 1971
47.7.113 ECC Error Injection Code Register (FR_EEICR)..... 1972
47.7.114 Message Buffer Configuration, Control, Status Register (FR_MBCCSR*n*)..... 1972
47.7.115 Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*)..... 1974
47.7.116 Message Buffer Frame ID Register (FR_MBFIDR*n*).....1976
47.7.117 Message Buffer Index Register (FR_MBIDXR*n*)..... 1976
47.7.118 Message Buffer Data Field Offset Register (FR_MBDOR*n*)..... 1977
47.7.119 LRAM ECC Error Test Register (FR_LEETR*n*).....1978
47.8 Functional Description.....1978
47.8.1 Message Buffer Concept..... 1978
47.8.2 Physical Message Buffer.....1978
47.8.3 Message Buffer Types..... 1980
47.8.4 FlexRay Memory Area Layout..... 1989
47.8.5 Physical Message Buffer Description..... 1993
47.8.6 Individual Message Buffer Functional Description..... 2003
47.8.7 Individual Message Buffer Search..... 2023
47.8.8 Individual Message Buffer Reconfiguration..... 2026

47.8.9 Receive FIFOs....2027
47.8.10 Channel Device Modes....2036
47.8.11 External Clock Synchronization....2038
47.8.12 Sync Frame ID and Sync Frame Deviation Tables....2039
47.8.13 MTS Generation....2043
47.8.14 Key Slot Transmission....2043
47.8.15 Sync Frame Filtering....2044
47.8.16 Strobe Signal Support....2046
47.8.17 Timer Support....2047
47.8.18 Slot Status Monitoring....2049
47.8.19 System Bus Access....2053
47.8.20 Interrupt Support....2055
47.8.21 Lower Bit Rate Support....2060
47.8.22 PE Data Memory (PE DRAM)....2061
47.8.23 CHI Lookup-Table Memory (CHI LRAM)....2063
47.8.24 Memory Content Fault Detection....2064
47.8.25 Memory Fault Injection....2069
47.8.26 StopWatch function....2072
47.9 Application Information....2074
47.9.1 Module Configuration....2074
47.9.2 Initialization Sequence....2075
47.9.3 Memory Fault Injection out of POC:default config....2077
47.9.4 Shut Down Sequence....2078
47.9.5 Number of Usable Message Buffers....2078
47.9.6 Protocol Control Command Execution....2079
47.9.7 Message Buffer Search On Simple Message Buffer Configuration....2080

Chapter 48
Synchronous Audio Interface (SAI)

48.1 Chip-specific SAI information....2085

48.1.1 SAI configuration....2085
48.1.2 SAI's stop-stop ack protocol....2086
48.2 Introduction....2086
48.2.1 Features....2086
48.2.2 Block diagram....2087
48.2.3 Modes of operation....2087
48.3 External signals....2088
48.4 Memory map and register definition....2089
48.4.1 SAI Transmit Control Register (I2S_TCSR)....2091
48.4.2 SAI Transmit Configuration 1 Register (I2S_TCR1)....2094
48.4.3 SAI Transmit Configuration 2 Register (I2S_TCR2)....2094
48.4.4 SAI Transmit Configuration 3 Register (I2S_TCR3)....2096
48.4.5 SAI Transmit Configuration 4 Register (I2S_TCR4)....2097
48.4.6 SAI Transmit Configuration 5 Register (I2S_TCR5)....2099
48.4.7 SAI Transmit Data Register (I2S_TDR*n*)....2100
48.4.8 SAI Transmit FIFO Register (I2S_TFR*n*)....2101
48.4.9 SAI Transmit Mask Register (I2S_TMR)....2101
48.4.10 SAI Receive Control Register (I2S_RCSR)....2102
48.4.11 SAI Receive Configuration 1 Register (I2S_RCR1)....2105
48.4.12 SAI Receive Configuration 2 Register (I2S_RCR2)....2106
48.4.13 SAI Receive Configuration 3 Register (I2S_RCR3)....2107
48.4.14 SAI Receive Configuration 4 Register (I2S_RCR4)....2108
48.4.15 SAI Receive Configuration 5 Register (I2S_RCR5)....2110
48.4.16 SAI Receive Data Register (I2S_RDR*n*)....2111
48.4.17 SAI Receive FIFO Register (I2S_RFR*n*)....2112
48.4.18 SAI Receive Mask Register (I2S_RMR)....2112
48.4.19 SAI MCLK Control Register (I2S_MCR)....2113
48.4.20 SAI MCLK Divide Register (I2S_MDR)....2114
48.5 Functional description....2115

48.5.1 SAI clocking ... 2115
48.5.2 SAI resets ... 2117
48.5.3 Synchronous modes ... 2118
48.5.4 Frame sync configuration ... 2119
48.5.5 Data FIFO ... 2120
48.5.6 Word mask register ... 2123
48.5.7 Interrupts and DMA requests ... 2124

Chapter 49
Periodic Interrupt Timer (PIT-RTI)

49.1 Chip-specific PIT-RTI information ... 2127
49.1.1 PIT configuration ... 2127
49.1.2 Input Output ... 2127
49.1.3 PIT-RTI's stop-stop ack protocol ... 2128
49.2 Introduction ... 2128
49.2.1 Block diagram ... 2128
49.2.2 Features ... 2129
49.3 Signal description ... 2130
49.4 Memory map/register description ... 2130
49.4.1 PIT Module Control Register (PIT_MCR) ... 2133
49.4.2 PIT Upper Lifetime Timer Register (PIT_LTMR64H) ... 2134
49.4.3 PIT Lower Lifetime Timer Register (PIT_LTMR64L) ... 2134
49.4.4 Timer Load Value Register (PIT_RTI_LDVAL) ... 2135
49.4.5 Current Timer Value Register (PIT_RTI_CVAL) ... 2135
49.4.6 Timer Control Register (PIT_RTI_TCTRL) ... 2136
49.4.7 Timer Flag Register (PIT_RTI_TFLG) ... 2137
49.4.8 Timer Load Value Register (PIT_LDVAL*n*) ... 2137
49.4.9 Current Timer Value Register (PIT_CVAL*n*) ... 2138
49.4.10 Timer Control Register (PIT_TCTRL*n*) ... 2138
49.4.11 Timer Flag Register (PIT_TFLG*n*) ... 2139

49.5 Functional description....2140
49.5.1 General operation....2140
49.5.2 Interrupts....2142
49.5.3 Chained timers....2142
49.6 Initialization and application information....2142
49.7 Example configuration for chained timers....2144
49.8 Example configuration for the lifetime timer....2144

Chapter 50
Software Watchdog Timer (SWT)

50.1 Chip-specific SWT information....2147
50.1.1 SWT configuration....2147
50.1.2 SWT register reset value....2148
50.1.3 Service mode watchpoint input....2148
50.2 Introduction....2148
50.2.1 Overview....2149
50.2.2 Features....2149
50.2.3 Modes of operation....2149
50.3 External signal description....2150
50.4 Memory Map and Registers....2150
50.4.1 SWT Control Register (SWT_CR)....2151
50.4.2 SWT Interrupt Register (SWT_IR)....2154
50.4.3 SWT Time-out Register (SWT_TO)....2155
50.4.4 SWT Window Register (SWT_WN)....2155
50.4.5 SWT Service Register (SWT_SR)....2156
50.4.6 SWT Counter Output Register (SWT_CO)....2156
50.4.7 SWT Service Key Register (SWT_SK)....2157
50.5 Functional description....2157
50.5.1 Introduction....2157
50.5.2 Configuration locking....2158

50.5.3 Unlock sequence 2159
50.5.4 Servicing operations 2159
50.5.5 Time-out 2161
50.5.6 Initialization 2161

Chapter 51
Real Time Clock (RTC-API)

51.1 Chip-specific RTC-API information 2163
51.1.1 RTC Configuration 2163
51.1.2 RTC register reset value 2163
51.1.3 Seamless RTC operation across functional and destructive resets 2163
51.1.4 RTC-API's stop-stop ack protocol 2164
51.2 Introduction 2164
51.2.1 Overview 2164
51.2.2 Features 2164
51.2.3 Block Diagram 2165
51.2.4 Modes of Operation 2166
51.3 Memory Map and Registers 2167
51.3.1 RTC Supervisor Control register (RTC_RTCSUPV) 2167
51.3.2 RTC Control register (RTC_RTCC) 2168
51.3.3 RTC Status register (RTC_RTCS) 2170
51.3.4 RTC Counter register (RTC_RTCCNT) 2172
51.3.5 API Compare Value register (RTC_APIVAL) 2172
51.3.6 RTC Compare Value register (RTC_RTCVAL) 2173
51.3.7 RTC Analogue comparator count register (RTC_ANLCMP_CNT) 2173
51.4 Functional Description 2173
51.4.1 RTC 2173
51.4.2 API functional description 2175

Chapter 52
System Timer Module (STM)

52.1 Chip-specific STM information....2177
52.1.1 STM configuration....2177
52.1.2 STM register reset value....2178
52.2 Introduction....2178
52.2.1 Overview....2178
52.2.2 Features....2178
52.2.3 Modes of operation....2178
52.3 External signal description....2179
52.4 Memory map and registers....2179
52.4.1 STM Control Register (STM_CR)....2180
52.4.2 STM Count Register (STM_CNT)....2181
52.4.3 STM Channel Control Register (STM_CCR*n*)....2182
52.4.4 STM Channel Interrupt Register (STM_CIR*n*)....2182
52.4.5 STM Channel Compare Register (STM_CMP*n*)....2183
52.5 Functional description....2183

Chapter 53
Reset Generation Module (MC_RGM)

53.1 Introduction....2185
53.1.1 Overview....2185
53.1.2 Features....2186
53.1.3 Reset Sources....2187
53.2 External Signal Description....2188
53.3 Memory Map and Register Definition....2189
53.3.1 'Destructive' Event Status Register (MC_RGM_DES)....2190
53.3.2 'Destructive' Bidirectional Reset Enable Register (MC_RGM_DBRE)....2192
53.3.3 'Functional' Event Status Register (MC_RGM_FES)....2194
53.3.4 'Functional' Event Reset Disable Register (MC_RGM_FERD)....2197
53.3.5 'Functional' Event Reset DisableAlternate Request Status Register (MC_RGM_FEAR)....2200
53.3.6 'Functional' Bidirectional Reset Enable Register (MC_RGM_FBRE)....2202

53.3.7 'Functional' Event Short Sequence Register (MC_RGM_FESS) 2205
53.3.8 'Functional' Reset Escalation Threshold Register (MC_RGM_FRET) 2208
53.3.9 'Destructive' Reset Escalation Threshold Register (MC_RGM_DRET) 2208
53.4 Functional Description 2209
53.4.1 Reset State Machine 2209
53.4.2 'Destructive' Resets 2213
53.4.3 External Reset 2213
53.4.4 'Functional' Resets 2214
53.4.5 Alternate Event Generation 2215
53.4.6 'Functional' Reset Escalation 2216
53.4.7 'Destructive' Reset Escalation 2217

Chapter 54
System Status and Configuration Module (SSCM)

54.1 Chip-specific SSCM information 2219
54.1.1 System Status and Configuration Module (SSCM) configuration 2219
54.2 Introduction 2220
54.2.1 Overview 2220
54.2.2 Glossary 2220
54.2.3 Features 2221
54.2.4 Modes of operation 2221
54.3 External signal description 2221
54.4 Memory map and register definition 2222
54.4.1 SSCM System Status (SSCM_STATUS) 2223
54.4.2 SSCM System Memory and ID Register (SSCM_MEMCONFIG) 2224
54.4.3 SSCM Debug Status Port Register (SSCM_DEBUGPORT) 2225
54.4.4 SSCM HSM and User Option Status Register (SSCM_UOPS) 2226
54.4.5 Processor Start Address Register (SSCM_PSA) 2227
54.4.6 SSCM HSM Start Address Register (SSCM_HSA) 2227
54.4.7 Life Cycle Status Register (SSCM_LCSTAT) 2228

54.5 Functional description....2228
54.5.1 DCF mechanism....2229
54.5.2 Boot mode functionality....2231
54.5.3 BAF configuration....2231
54.5.4 HSM boot header search....2231
54.5.5 Life Cycle....2232
54.6 Initialization and application information....2234
54.6.1 Reset....2234
54.7 Additional safety measures....2234
54.7.1 Spurious reset protection....2234

Chapter 55
Boot Assist Flash (BAF)

55.1 Chip-specific BAF information....2235
55.1.1 BAF configuration....2235
55.1.2 BAF memory map....2235
55.1.3 LinFLEXD_0 pin configuration....2235
55.1.4 Search boot header and boot options....2236
55.1.5 MPC5746C cores/BAF CPUn....2237
55.1.6 BAF NMI event handling....2238
55.2 Introduction....2239
55.3 BAF image header....2240
55.3.1 BAF image version....2240
55.4 Functional description....2240
55.4.1 Boot modes....2240
55.4.2 Device initialization....2241
55.4.3 Flow of control....2242
55.4.4 Optionally remap calibration RAM....2243
55.4.5 Optionally wait for HSM....2243
55.4.6 BAF SAFE mode event handling....2244

55.4.7 Initialization of BAF DCF clients....2245
55.4.8 Optionally perform serial boot....2254
55.4.9 Serial boot configuration....2258
55.5 Resources....2260

Chapter 56
Nexus Crossbar Multi-Master Client (NXMC)

56.1 Chip-specific NXMC information....2263
56.1.1 NXMC block connections....2263
56.1.2 NXMC source and master client IDs....2263
56.2 Introduction....2264
56.2.1 Features....2264
56.3 External signal description....2265
56.4 Supported messages and TCODEs....2266
56.5 Register description....2269
56.5.1 Development control register 1 (DC1)....2269
56.5.2 Development control register 2 (DC2)....2270
56.5.3 Watchpoint trigger register (WT)....2271
56.5.4 Data trace control register (DTC)....2272
56.5.5 Data trace start address registers 1 and 2 (DTSA1 and DTSA2)....2273
56.5.6 Data trace end address registers 1 and 2 (DTEA1 and DTEA2)....2273
56.5.7 Breakpoint/watchpoint control register 1 (BWC1)....2274
56.5.8 Breakpoint/watchpoint control register 2 (BWC2)....2275
56.5.9 Breakpoint/watchpoint address registers 1 and 2 (BWA1 and BWA2)....2276
56.5.10 Unimplemented registers....2276
56.6 Programming considerations (RESET)....2277
56.6.1 IEEE 1149.1 (JTAG) Test Access Port....2277
56.6.2 Enabling NXMC operation....2277
56.6.3 Enabling NXMC TAP controller....2277
56.6.4 NXMC register access via JTAG....2278

56.7 Functional description....2279
56.7.1 Data trace.... 2279
56.7.2 Watchpoint support.... 2285
56.7.3 Timestamping feature.... 2287

Chapter 57
JTAG Controller (JTAGC)

57.1 Chip-specific JTAG information.... 2289
57.1.1 JTAG Controller (JTAGC) configuration....2289
57.1.2 JTAGC Control 1 register.... 2289
57.2 Introduction....2290
57.2.1 Block diagram.... 2291
57.2.2 Features.... 2291
57.2.3 Modes of operation.... 2292
57.3 External signal description....2293
57.3.1 TCK—Test clock input....2293
57.3.2 TDI—Test data input.... 2293
57.3.3 TDO—Test data output....2294
57.3.4 TMS—Test mode select....2294
57.4 Register description.... 2294
57.4.1 Instruction register.... 2294
57.4.2 Bypass register.... 2295
57.4.3 Device identification register.... 2295
57.4.4 JTAG_PASSWORD register.... 2296
57.4.5 Boundary scan register....2296
57.5 Functional description....2296
57.5.1 JTAGC reset configuration.... 2297
57.5.2 IEEE 1149.1-2001 (JTAG) Test Access Port.... 2297
57.5.3 TAP controller state machine....2297
57.5.4 JTAGC block instructions....2299

57.5.5 Boundary scan....2302
57.6 Initialization/Application information....2303

Chapter 58
JTAG Data Communication (JDC)

58.1 Introduction....2305
58.2 Overview....2305
58.3 Memory mapped registers....2306
58.3.1 Module Configuration Register (JDC_MCR)....2307
58.3.2 Module Status Register (JDC_MSR)....2308
58.3.3 JTAG Output Data Register (JDC_JOUT_IPS)....2309
58.3.4 JTAG Input Data Register (JDC_JIN_IPS)....2310
58.4 Non-Memory mapped register definition....2310
58.4.1 JTAG output data register (JOUT)....2310
58.4.2 JTAG input data register (JIN)....2311
58.5 Functional description....2311
58.5.1 JTAG register access....2312
58.6 Use case example....2312

Chapter 59
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)

59.1 References....2315
59.2 Abbreviations....2315
59.3 Introduction....2316
59.3.1 Types of operation....2317
59.3.2 Deployment by class....2317
59.3.3 1149.7 TAP signals....2318
59.3.4 TAP.7 architecture....2319
59.3.5 Protocols....2320
59.4 Operating models....2322
59.5 CJTAG implementation summary....2322

59.5.1 T0 functions ... 2322
59.5.2 T1 functions ... 2323
59.5.3 T2 functions ... 2323
59.5.4 T3 functions ... 2323
59.5.5 T4 functions ... 2323
59.6 Ancillary services ... 2323
59.6.1 Overview ... 2323
59.6.2 Resets ... 2324
59.6.3 Start-up Options ... 2327
59.6.4 RSU operation ... 2328
59.6.5 TAPC State Machine ... 2330
59.7 EPU (Extended Protocol Unit) Operation ... 2331
59.7.1 EPU Operation ... 2331
59.7.2 EPU Registers ... 2334
59.7.3 EPU Commands ... 2345
59.7.4 EPU operating states ... 2353
59.7.5 System and EPU Paths ... 2354
59.8 APU (Advanced Protocol Unit) Operation ... 2354
59.8.1 Overview ... 2354
59.8.2 Operation ... 2357
59.8.3 Escape sequences ... 2359
59.8.4 Signal behaviors ... 2359
59.8.5 APU Functions ... 2360
59.8.6 Configuration Change Packets (CP) ... 2366
59.8.7 Scan Packet ... 2367
59.8.8 SP Format ... 2367
59.9 Functional Description ... 2374
59.9.1 Switching from Standard Protocol to Advanced Protocol ... 2374

Chapter 60

Nexus Port Controller (NPC)

60.1 Chip-specific NPC information ... 2375
60.1.1 NPC configuration ... 2375
60.1.2 EVTO sharing - Pulse time ... 2376
60.1.3 Trace messages for transactions with error responses ... 2376
60.1.4 Trace loss during resets ... 2376
60.1.5 EVTI_IN pin ... 2377
60.2 Introduction ... 2377
60.2.1 Overview ... 2377
60.2.2 Features ... 2378
60.2.3 Modes of operation ... 2378
60.3 External signal description ... 2380
60.3.1 Overview ... 2380
60.3.2 Detailed signal descriptions ... 2380
60.4 Register definition ... 2382
60.4.1 Register descriptions ... 2382
60.5 Functional Description ... 2386
60.5.1 NPC reset configuration ... 2386
60.5.2 Auxiliary Output Port ... 2386
60.5.3 IEEE 1149.1-2001 (JTAG) TAP ... 2389
60.5.4 Nexus JTAG Port Sharing ... 2393
60.5.5 MCKO Control ... 2393
60.5.6 EVTO Sharing ... 2394
60.5.7 System Clock Locked Indication ... 2394
60.6 Initialization/Application Information ... 2394
60.6.1 Accessing NPC tool-mapped registers ... 2394

Chapter 61
Core Complex Overview

61.1 Overview of the e200z210n3 and e200z4204n3 cores ... 2397

61.2 z210n3 microarchitecture summary....2397
61.2.1 Overview of the e200z210n3 core....2398
61.2.2 z210n3 instruction unit features....2400
61.2.3 z210n3 integer unit features....2400
61.2.4 z210n3 load/store unit features....2400
61.2.5 z210n3 performance monitor features....2401
61.2.6 z210n3 system bus features....2401
61.3 z4204n3 microarchitecture summary....2401
61.3.1 Overview of the e200z4204n3....2403
61.3.2 z4204n3 instruction unit features....2403
61.3.3 z4204n3 integer unit features....2404
61.3.4 z210n3 load/store unit features....2404
61.3.5 z4204n3 EFPU2 floating-point unit features....2404
61.3.6 z4204n3 cache features....2405
61.3.7 z4204n3 performance monitor features....2406
61.3.8 z4204n3 system bus features....2407

Chapter 62
Z2 Core description

62.1 Overview of the e200z210n3 core....2409
62.2 Register model....2409
62.2.1 Branch Unit Control and Status Register (BUCSR)....2412
62.2.2 Hardware Implementation Dependent Register 0 (HID0)....2414
62.2.3 Hardware Implementation Dependent Register 1 (HID1)....2415
62.3 Single-issue operation....2417
62.4 Instruction timing....2417
62.5 Exceptions....2424
62.5.1 Exception Syndrome Register (ESR)....2425
62.5.2 Machine State Register (MSR)....2426
62.5.3 Machine Check Syndrome Register (MCSR)....2428

62.5.4 Interrupt Vector Prefix Registers (IVPR) 2431
62.5.5 Interrupt Definitions 2432
62.6 End-to-End ECC Support 2443
62.6.1 End-to-End ECC control and configuration 2443

Chapter 63
Z4 Core description

63.1 Overview of the e200z4204n3 core 2447
63.2 Register model 2447
63.2.1 Hardware Implementation Dependent Register 0 (HID0) 2451
63.2.2 Hardware Implementation Dependent Register 1 (HID1) 2453
63.3 Instruction timing 2454
63.4 Dual-issue operation 2461
63.5 Reservation instructions and cache interactions 2462
63.6 Signal Processing Extension/Embedded Floating-point Status and Control Register (SPEFSCR) 2462
63.7 Cache 2465
63.7.1 Cache overview 2465
63.7.2 L1 Cache Control and Status Register 0 (L1CSR0) 2465
63.7.3 L1 Cache Control and Status Register 1 (L1CSR1) 2468
63.7.4 L1 Cache Configuration Register 0 (L1CFG0) 2471
63.7.5 L1 Cache Configuration Register 1 (L1CFG1) 2472
63.7.6 Data cache software coherency 2472
63.7.7 Data cache hardware coherency 2473
63.7.8 Cache invalidate by set and way 2473
63.7.9 Cache EDC/ECC parity protection 2475
63.7.10 Cache error injection 2477
63.8 Exceptions 2478
63.8.1 Exception Syndrome Register (ESR) 2478
63.8.2 Machine State Register (MSR) 2480
63.8.3 Machine Check Syndrome Register (MCSR) 2482

63.8.4 Interrupt Vector Prefix Registers (IVPR)..... 2485
63.8.5 Interrupt definitions.....2485
63.9 End-to-End ECC support..... 2496
63.9.1 End-to-End ECC control and configuration.....2496

Chapter 64
Z2 Core Debug Support

64.1 Overview.....2501
64.1.1 Software Debug Facilities.....2501
64.1.2 Additional Debug Facilities..... 2502
64.1.3 Hardware Debug Facilities.....2503
64.1.4 Sharing Debug Resources by Software/Hardware.....2503
64.2 Software Debug Events and Exceptions.....2505
64.2.1 Instruction Address Compare Event..... 2507
64.2.2 Data Address Compare Event..... 2507
64.2.3 Linked Instruction Address Compare and Data Address Compare Events..... 2511
64.2.4 Trap Debug Event..... 2512
64.2.5 Branch Taken Debug Event..... 2512
64.2.6 Instruction Complete Debug Event.....2512
64.2.7 Interrupt Taken Debug Event.....2513
64.2.8 Critical Interrupt Taken Debug Event.....2513
64.2.9 Return Debug Event.....2513
64.2.10 Critical Return Debug Event.....2514
64.2.11 External debug event.....2514
64.2.12 Unconditional debug event..... 2514
64.2.13 Performance Monitor Interrupt debug event.....2515
64.3 Debug registers..... 2515
64.3.1 Debug Address and Value Registers.....2516
64.3.2 Debug Control and Status registers.....2517
64.3.3 External Debug Resource Allocation Control Register (EDBRAC0)..... 2540

64.3.4 Debug Event Select Register (DEVENT)....2547
64.3.5 Debug Data Acquisition Message Register (DDAM)....2548
64.4 Using Debug Resources for Stack Limit Checking....2549
64.5 External Debug Support....2551
64.5.1 External Debug Registers....2553
64.5.2 OnCE Introduction....2559
64.5.3 JTAG/OnCE Pins....2561
64.5.4 OnCE Internal Interface Signals....2562
64.5.5 OnCE Interface Signals....2562
64.5.6 OnCE Controller and serial interface....2565
64.5.7 Access to Debug Resources....2572
64.5.8 Methods of Entering Debug Mode....2574
64.5.9 CPU Status and Control Scan Chain Register (CPUSCR)....2576
64.5.10 Reserved Registers (Reserved)....2583
64.6 Watchpoint Support....2583
64.7 Basic Steps for Enabling, Using, and Exiting External Debug Mode....2586

Chapter 65
Z4 Core Debug Support

65.1 Overview....2589
65.1.1 Software Debug Facilities....2589
65.1.2 Additional Debug Facilities....2590
65.1.3 Hardware Debug Facilities....2591
65.1.4 Sharing Debug Resources by Software/Hardware....2591
65.2 Software Debug Events and Exceptions....2593
65.2.1 Instruction Address Compare Event....2595
65.2.2 Data Address Compare Event....2595
65.2.3 Linked Instruction Address Compare and Data Address Compare Events....2604
65.2.4 Trap Debug Event....2605
65.2.5 Branch Taken Debug Event....2605

65.2.6 Instruction Complete Debug Event....2606
65.2.7 Interrupt Taken Debug Event....2606
65.2.8 Critical Interrupt Taken Debug Event....2607
65.2.9 Return Debug Event....2607
65.2.10 Critical Return Debug Event....2607
65.2.11 External debug event....2608
65.2.12 Unconditional debug event....2608
65.2.13 Performance Monitor Interrupt debug event....2608
65.3 Debug registers....2609
65.3.1 Debug Address and Value Registers....2609
65.3.2 Debug Control and Status registers....2610
65.3.3 External Debug Resource Allocation Control Register (EDBRAC0)....2633
65.3.4 Debug Event Select Register (DEVENT)....2640
65.3.5 Debug Data Acquisition Message Register (DDAM)....2641
65.4 Using Debug Resources for Stack Limit Checking....2642
65.5 External Debug Support....2645
65.5.1 External Debug Registers....2646
65.5.2 OnCE Introduction....2652
65.5.3 JTAG/OnCE Pins....2655
65.5.4 OnCE Internal Interface Signals....2655
65.5.5 OnCE Interface Signals....2656
65.5.6 OnCE Controller and serial interface....2659
65.5.7 Access to Debug Resources....2666
65.5.8 Methods of Entering Debug Mode....2668
65.5.9 CPU Status and Control Scan Chain Register (CPUSCR)....2670
65.5.10 Reserved Registers (Reserved)....2677
65.6 Watchpoint Support....2677
65.7 Cache Array Access During Debug....2680
65.8 Basic Steps for Enabling, Using, and Exiting External Debug Mode....2680

Chapter 66
Z2 Nexus 3 Module

66.1 Introduction....2683
66.1.1 General description....2683
66.1.2 Terms and definitions....2683
66.1.3 Feature list....2684
66.1.4 Functional block diagram....2686
66.2 Enabling Nexus 3 operation....2687
66.2.1 Interaction with Low Power Modes....2688
66.3 TCODEs supported....2689
66.4 Nexus 3 Programmer's model....2694
66.4.1 Client Select Control (CSC) register....2696
66.4.2 Port Configuration Register (PCR) - reference only....2696
66.4.3 Nexus Development Control Register 1 (DC1)....2697
66.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)....2699
66.4.5 Nexus Development Control Register 4 (DC4)....2701
66.4.6 Development Status Register (DS)....2702
66.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers....2703
66.4.8 Nexus Watchpoint Mask (WMSK) register....2710
66.4.9 Nexus Overrun Control Register (OVCR)....2711
66.4.10 Data Trace Control Register (DTC) register....2712
66.4.11 Data Trace Start Address Registers (DTSA1-4)....2714
66.4.12 Data Trace End Address (DTEA1-4) registers....2715
66.4.13 Read/Write Access Control/Status (RWCS) register....2716
66.4.14 Read/Write Access Data (RWD) register....2718
66.4.15 Read/Write Access Address (RWA)....2719
66.5 Nexus 3 Register Access via JTAG/OnCE....2720
66.6 Nexus 3 Register Access via Software....2720
66.7 Nexus message fields....2721

66.7.1 TCODE Field ... 2721
66.7.2 Source ID Field (SRC) ... 2721
66.7.3 Relative Address Field (U-ADDR) ... 2721
66.7.4 Full Address Field (F-ADDR) ... 2722
66.7.5 Timestamp field (TSTAMP) ... 2723
66.8 Nexus Message Queues ... 2723
66.8.1 Message Queue Overrun ... 2724
66.8.2 CPU Stall ... 2724
66.8.3 Message Suppression ... 2724
66.8.4 Nexus Message Priority ... 2725
66.8.5 Data Acquisition Message Priority Loss Response ... 2726
66.8.6 Ownership Trace Message Priority Loss Response ... 2726
66.8.7 Program Trace Message Priority Loss Response ... 2727
66.8.8 Program Trace Sync Message Priority Loss Response ... 2727
66.8.9 Data Trace Message Priority Loss Response ... 2727
66.9 Debug Status messages ... 2727
66.10 Error messages ... 2728
66.11 Ownership trace ... 2728
66.11.1 Overview ... 2728
66.11.2 Ownership Trace Messaging (OTM) ... 2728
66.12 Program trace ... 2730
66.12.1 Branch Trace Messaging types ... 2730
66.12.2 BTM Message For mats ... 2733
66.12.3 Program Trace Message Fields ... 2734
66.12.4 Resource Full Messages ... 2735
66.12.5 Program Correlation Messages ... 2736
66.12.6 Program Trace Overflow Error messages ... 2738
66.12.7 Program Trace Synchronization messages ... 2738
66.12.8 Program Trace Direct/Indirect Branch with Sync messages ... 2740

66.12.9 Enabling Program Trace ... 2741
66.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration) ... 2741
66.13 Data Trace ... 2743
66.13.1 Data Trace Messaging (DTM) ... 2743
66.13.2 DTM Message formats ... 2743
66.13.3 DTM Operation ... 2747
66.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration) ... 2748
66.14 Data Acquisition messaging ... 2749
66.14.1 Data Acquisition ID Tag field ... 2749
66.14.2 Data Acquisition Data field ... 2750
66.14.3 Data Acquisition Trace event ... 2750
66.15 Watchpoint Trace messaging ... 2750
66.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration) ... 2752
66.16 Nexus 3 Read/Write access to memory-mapped resources ... 2753
66.16.1 Single write access ... 2753
66.16.2 Block write access ... 2754
66.16.3 Single read access ... 2755
66.16.4 Block read access ... 2756
66.16.5 Error handling ... 2757
66.16.6 Read/Write access error message ... 2758
66.17 Nexus 3 pin interface ... 2759
66.17.1 Pins implemented ... 2759
66.17.2 Pin protocol ... 2761
66.18 Rules for output messages ... 2763
66.19 Auxiliary port arbitration ... 2764
66.20 Examples ... 2764
66.21 Electrical characteristics ... 2767
66.22 IEEE 1149.1 (JTAG) RD/WR sequences ... 2767
66.22.1 JTAG sequence for accessing internal Nexus registers ... 2768

66.22.2 JTAG sequence for read access of memory-mapped resources....2768
66.22.3 JTAG sequence for write access of memory-mapped resources....2768

Chapter 67
Z4 Nexus 3 Module

67.1 Introduction....2771
67.1.1 General Description....2771
67.1.2 Terms and Definitions....2771
67.1.3 Feature List....2772
67.1.4 Functional Block Diagram....2774
67.2 Enabling Nexus 3 Operation....2775
67.2.1 Interaction with Low Power Modes....2776
67.3 TCODEs supported....2777
67.4 Nexus 3 Programmer's Model....2782
67.4.1 Client Select Control (CSC) register....2783
67.4.2 Port Configuration Register (PCR) - reference only....2784
67.4.3 Nexus Development Control Register 1 (DC1)....2785
67.4.4 Nexus Development Control Registers 2 and 3 (DC2, DC3)....2786
67.4.5 Nexus Development Control Register 4 (DC4)....2788
67.4.6 Development Status Register (DS)....2789
67.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers....2790
67.4.8 Nexus Watchpoint Mask (WMSK) register....2797
67.4.9 Nexus Overrun Control Register (OVCR)....2798
67.4.10 Data Trace Control Register (DTC) register....2799
67.4.11 Data Trace Start Address Registers (DTSA1-4)....2802
67.4.12 Data Trace End Address (DTEA1-4) registers....2802
67.4.13 Read/Write Access Control/Status (RWCS) register....2803
67.4.14 Read/Write Access Data (RWD) register....2805
67.4.15 Read/Write Access Address (RWA)....2807
67.5 Nexus 3 Register Access via JTAG/OnCE....2807

67.6 Nexus 3 Register Access via Software ... 2808
67.7 Nexus Message Fields ... 2808
67.7.1 TCODE Field ... 2808
67.7.2 Source ID Field (SRC) ... 2808
67.7.3 Relative Address Field (U-ADDR) ... 2809
67.7.4 Full Address Field (F-ADDR) ... 2810
67.8 Nexus Message Queues ... 2810
67.8.1 Message Queue Overrun ... 2811
67.8.2 CPU Stall ... 2811
67.8.3 Message Suppression ... 2811
67.8.4 Nexus Message Priority ... 2812
67.8.5 Data Acquisition Message Priority Loss Response ... 2813
67.8.6 Ownership Trace Message Priority Loss Response ... 2813
67.8.7 Program Trace Message Priority Loss Response ... 2814
67.8.8 Program Trace Sync Message Priority Loss Response ... 2814
67.8.9 Data Trace Message Priority Loss Response ... 2814
67.9 Debug Status messages ... 2814
67.10 Error messages ... 2815
67.11 Ownership trace ... 2815
67.11.1 Overview ... 2815
67.11.2 Ownership Trace Messaging (OTM) ... 2815
67.12 Program trace ... 2817
67.12.1 Branch Trace Messaging types ... 2817
67.12.2 BTM Message For mats ... 2820
67.12.3 Program Trace Message Fields ... 2821
67.12.4 Resource Full Messages ... 2822
67.12.5 Program Correlation Messages ... 2823
67.12.6 Program Trace Overflow Error messages ... 2825
67.12.7 Program Trace Synchronization messages ... 2825

67.12.8 Program Trace Direct/Indirect Branch with Sync messages....2827
67.12.9 Enabling Program Trace....2828
67.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)....2828
67.13 Data Trace....2830
67.13.1 Data Trace Messaging (DTM)....2830
67.13.2 DTM Message formats....2830
67.13.3 DTM Operation....2834
67.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)....2835
67.14 Data Acquisition messaging....2836
67.14.1 Data Acquisition ID Tag field....2836
67.14.2 Data Acquisition Data field....2836
67.14.3 Data Acquisition Trace event....2836
67.15 Watchpoint Trace messaging....2837
67.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)....2839
67.16 Nexus 3 Read/Write access to memory-mapped resources....2839
67.16.1 Single write access....2840
67.16.2 Block write access....2841
67.16.3 Single read access....2842
67.16.4 Block read access....2843
67.16.5 Error handling....2844
67.16.6 Read/Write access error message....2845
67.17 Nexus 3 pin interface....2845
67.17.1 Pins implemented....2846
67.17.2 Pin protocol....2848
67.18 Rules for output messages....2850
67.19 Auxiliary port arbitration....2851
67.20 Examples....2851
67.21 Electrical characteristics....2854
67.22 IEEE 1149.1 (JTAG) RD/WR sequences....2854

67.22.1 JTAG sequence for accessing internal Nexus registers.... 2855
67.22.2 JTAG sequence for read access of memory-mapped resources.... 2855
67.22.3 JTAG sequence for write access of memory-mapped resources.... 2855

Chapter 68
Enhanced Direct Memory Access (eDMA)

68.1 Introduction.... 2857
68.1.1 eDMA system block diagram.... 2857
68.1.2 Block parts.... 2858
68.1.3 Features.... 2859
68.2 Modes of operation.... 2860
68.3 Memory map/register definition.... 2861
68.3.1 TCD memory.... 2861
68.3.2 TCD initialization.... 2861
68.3.3 TCD structure.... 2861
68.3.4 Reserved memory and bit fields.... 2862
68.3.5 Control Register (DMA_CR).... 2889
68.3.6 Error Status Register (DMA_ES).... 2892
68.3.7 Enable Request Register (DMA_ERQ).... 2895
68.3.8 Enable Error Interrupt Register (DMA_EEI).... 2898
68.3.9 Set Enable Request Register (DMA_SERQ).... 2902
68.3.10 Clear Enable Request Register (DMA_CERQ).... 2903
68.3.11 Set Enable Error Interrupt Register (DMA_SEEI).... 2904
68.3.12 Clear Enable Error Interrupt Register (DMA_CEEI).... 2905
68.3.13 Clear Interrupt Request Register (DMA_CINT).... 2906
68.3.14 Clear Error Register (DMA_CERR).... 2907
68.3.15 Set START Bit Register (DMA_SSRT).... 2908
68.3.16 Clear DONE Status Bit Register (DMA_CDNE).... 2909
68.3.17 Interrupt Request Register (DMA_INT).... 2910
68.3.18 Error Register (DMA_ERR).... 2913

68.3.19 Hardware Request Status Register (DMA_HRS)..... 2917
68.3.20 Channel n Priority Register (DMA_DCHPRI*n*)..... 2923
68.3.21 Channel n Master ID Register (DMA_DCHMID*n*)..... 2924
68.3.22 TCD Source Address (DMA_TCD*n*_SADDR)..... 2925
68.3.23 TCD Transfer Attributes (DMA_TCD*n*_ATTR)..... 2925
68.3.24 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF)..... 2926
68.3.25 TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD*n*_NBYTES_MLNO)..... 2927
68.3.26 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO)..... 2927
68.3.27 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES)..... 2929
68.3.28 TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST)..... 2930
68.3.29 TCD Destination Address (DMA_TCD*n*_DADDR)..... 2930
68.3.30 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_CITER_ELINKYES)..... 2931
68.3.31 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO)..... 2932
68.3.32 TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF)..... 2933
68.3.33 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA)..... 2933
68.3.34 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES)..... 2934
68.3.35 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO)..... 2935
68.3.36 TCD Control and Status (DMA_TCD*n*_CSR)..... 2936
68.4 Functional description..... 2938
68.4.1 eDMA basic data flow..... 2938
68.4.2 Fault reporting and handling..... 2941
68.4.3 Channel preemption..... 2944
68.5 Initialization/application information..... 2944
68.5.1 eDMA initialization..... 2944

68.5.2 Programming errors ... 2946
68.5.3 Arbitration mode considerations ... 2947
68.5.4 Performing DMA transfers ... 2948
68.5.5 Monitoring transfer descriptor status ... 2952
68.5.6 Channel Linking ... 2954
68.5.7 Dynamic programming ... 2955

Chapter 69
Direct Memory Access Multiplexer

69.1 Chip-specific DMA MUX information ... 2961
69.1.1 DMA MUX configuration ... 2961
69.2 Introduction ... 2963
69.2.1 Overview ... 2963
69.2.2 Features ... 2964
69.2.3 Modes of operation ... 2964
69.3 External signal description ... 2965
69.4 Memory map/register definition ... 2965
69.4.1 Channel Configuration register (DMAMUX_CHCFG*n*) ... 2966
69.5 Functional description ... 2967
69.5.1 DMA channels with periodic triggering capability ... 2967
69.5.2 DMA channels with no triggering capability ... 2969
69.5.3 Always-enabled DMA sources ... 2970
69.6 Initialization/application information ... 2971
69.6.1 Reset ... 2971
69.6.2 Enabling and configuring sources ... 2971

Chapter 70
PRAMC

70.1 Chip-specific PRAMC information ... 2975
70.1.1 Register reset value ... 2975
70.2 Introduction ... 2975

70.3 SRAM controller memory map and register definition....2977
70.3.1 Platform RAM Configuration Register 1 (PRAMC_PRCR1)....2977
70.4 Functional description....2979
70.4.1 Read and Write operations....2979
70.4.2 Writes....2980
70.5 Initialization/application information....2982
70.6 Reliability considerations....2982
70.6.1 Hsiao ECC algorithm....2982
70.6.2 Transaction monitor....2985

Chapter 71
Flash memory controller

71.1 Chip-specific Flash memory controller information....2987
71.1.1 Master IDs....2987
71.2 Introduction....2987
71.3 Features....2988
71.4 Block diagrams....2988
71.5 Flash memory controller memory map....2989
71.5.1 Platform Flash Configuration Register 1 (PFLASH_PFCR1)....2993
71.5.2 Platform Flash Configuration Register 2 (PFLASH_PFCR2)....2996
71.5.3 Platform Flash Configuration Register 3 (PFLASH_PFCR3)....2999
71.5.4 Platform Flash Access Protection Register (PFLASH_PFAPR)....3001
71.5.5 Platform Flash Remap Control Register (PFLASH_PFCRCR)....3004
71.5.6 Platform Flash Remap Descriptor Enable Register (PFLASH_PFCRDE)....3006
71.5.7 Platform Flash Configuration Register 4 (PFLASH_PFCR4)....3009
71.5.8 Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD*n*_Word0)....3012
71.5.9 Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD*n*_Word1)....3013
71.5.10 Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD*n*_Word2)....3014
71.6 Functional description....3019
71.6.1 Basic interface protocol....3020

71.6.2 Access protections....3020
71.6.3 Security module exclusive control....3022
71.6.4 Security Module Alternate Programming Interface....3022
71.6.5 Access pipelining....3023
71.6.6 Line read buffers and prefetch operation....3024
71.6.7 Instruction/Data prefetch triggering....3025
71.6.8 Per-Master prefetch triggering....3025
71.6.9 Buffer allocation....3025
71.6.10 PFlash calibration remap support....3026
71.6.11 Reliability considerations....3028
71.6.12 ECC on data flash accesses....3033
71.6.13 Array integrity considerations....3034

Chapter 72
Embedded Flash Memory

72.1 Chip-specific Embedded Flash memory information....3035
72.1.1 Embedded Flash Memory configuration....3035
72.1.2 Register bit mapping....3035
72.1.3 C55FMC_LOCK0 and C55FMC_SEL0 register bit mapping....3035
72.1.4 C55FMC_LOCK1 and C55FMC_SEL1 register bit mapping....3036
72.1.5 C55FMC_LOCK2 and C55FMC_SEL2 register bit mapping....3037
72.1.6 C55FMC_LOCK3 and C55FMC_SEL3 register bit mapping....3038
72.1.7 Embedded Flash Memory register reset value....3039
72.2 Introduction....3039
72.2.1 Overview....3040
72.2.2 Features....3041
72.2.3 Modes of operation....3041
72.3 UTest NVM block....3042
72.4 Test mode disable seal....3042
72.5 Memory map and register definition....3043

72.5.1 Module Configuration Register (C55FMC_MCR)....3045
72.5.2 Extended Module Configuration Register (C55FMC_MCRE)....3050
72.5.3 Lock 0 register (C55FMC_LOCK0)....3053
72.5.4 Lock 1 register (C55FMC_LOCK1)....3055
72.5.5 Lock 2 register (C55FMC_LOCK2)....3055
72.5.6 Lock 3 register (C55FMC_LOCK3)....3056
72.5.7 Select 0 register (C55FMC_SEL0)....3057
72.5.8 Select 1 register (C55FMC_SEL1)....3058
72.5.9 Select 2 register (C55FMC_SEL2)....3059
72.5.10 Select 3 register (C55FMC_SEL3)....3060
72.5.11 Address register (C55FMC_ADR)....3061
72.5.12 UTest 0 register (C55FMC_UT0)....3063
72.5.13 UMISR register (C55FMC_UM*n*)....3066
72.5.14 UMISR register (C55FMC_UM9)....3067
72.5.15 Over-Program Protection 0 register (C55FMC_OPP0)....3068
72.5.16 Over-Program Protection 1 register (C55FMC_OPP1)....3069
72.5.17 Over-Program Protection 2 register (C55FMC_OPP2)....3070
72.5.18 Over-Program Protection 3 register (C55FMC_OPP3)....3070
72.5.19 Test Mode Disable Password Check register (C55FMC_TMD)....3071
72.6 Functional Description....3072
72.6.1 User Mode....3073
72.6.2 Low Power mode....3081
72.6.3 UTest mode....3081
72.7 Initialization information....3086

Chapter 73
Memory Error Management Unit (MEMU)

73.1 Chip-specific MEMU information....3087
73.1.1 MEMU configuration....3087
73.1.2 MEMU system connections....3087

73.1.3 MEMU error sources ... 3088
73.1.4 Reporting table implementation ... 3088
73.1.5 Concurrent overflow register (OFLW) implementation for MEMU_0 ... 3089
73.1.6 Concurrent overflow register (OFLW) implementation for MEMU_1 ... 3091
73.1.7 MEMU_0 registers ... 3092
73.1.8 MEMU_1 registers ... 3094
73.2 Introduction ... 3095
73.3 Features ... 3096
73.4 Block diagram ... 3097
73.5 Design overview ... 3097
73.6 External signal description ... 3101
73.7 Memory map and register definition ... 3102
73.7.1 Overview ... 3102
73.7.2 Control register (MEMU_CTRL) ... 3106
73.7.3 Error flag register (MEMU_ERR_FLAG) ... 3106
73.7.4 Debug register (MEMU_DEBUG) ... 3110
73.7.5 System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS*n*) ... 3112
73.7.6 System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR*n*) ... 3113
73.7.7 System RAM uncorrectable error reporting table status register (MEMU_SYS_RAM_UNCERR_STS) ... 3113
73.7.8 System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR) ... 3114
73.7.9 System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW*n*) ... 3114
73.7.10 Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS*n*) ... 3115
73.7.11 Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR*n*) ... 3116
73.7.12 Peripheral RAM uncorrectable error reporting table status register (MEMU_PERIPH_RAM_UNCERR_STS) ... 3116
73.7.13 Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR) ... 3117

73.7.14 Peripheral RAM concurrent overflow register (MEMU_PERIPH_RAM_OFLW*n*)..... 3117
73.7.15 Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS*n*)..... 3118
73.7.16 Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR*n*)..... 3119
73.7.17 Flash memory uncorrectable error reporting table status register (MEMU_FLASH_UNCERR_STS).....3119
73.7.18 Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR)..... 3120
73.7.19 Flash memory concurrent overflow register (MEMU_FLASH_OFLW*n*)..... 3120
73.8 Functional description.....3121
73.8.1 Initializing MEMU.....3121
73.8.2 Reading the reporting table..... 3122
73.8.3 Handling overflows (Multiple error reporting).....3122

Chapter 74
Fault Collection and Control Unit (FCCU)

74.1 Chip-specific FCCU information.....3125
74.1.1 FCCU configuration.....3125
74.1.2 FCCU register reset values 3125
74.1.3 FCCU_CFG register event bit values by source (N and C).....3126
74.1.4 Fault inputs.....3128
74.1.5 Error signal flow diagram..... 3133
74.2 Introduction.....3134
74.2.1 Acronyms and abbreviations.....3135
74.3 Main features..... 3135
74.4 Block diagram.....3136
74.5 Signal description.....3137
74.5.1 Signals.....3137
74.6 Put FCCU in Configuration or Normal state..... 3138
74.6.1 Introduction.....3138
74.6.2 About changing states..... 3139
74.6.3 Put FCCU in Configuration state..... 3139

74.6.4 Put FCCU in Normal state ... 3140
74.7 Run operations ... 3140
74.7.1 Introduction ... 3140
74.7.2 About running operations ... 3140
74.7.3 Run an operation ... 3141
74.8 Lock and unlock the configuration ... 3141
74.8.1 Introduction ... 3141
74.8.2 About locking the configuration ... 3141
74.8.3 Temporarily lock the configuration ... 3141
74.8.4 Temporarily unlock the configuration ... 3142
74.8.5 Permanently lock the configuration ... 3142
74.9 Functional description ... 3142
74.9.1 Definitions ... 3142
74.9.2 FSM description ... 3143
74.9.3 Fault priority scheme and nesting ... 3145
74.9.4 Fault recovery ... 3146
74.9.5 NMI/WKPU interface ... 3148
74.9.6 STCU interface ... 3149
74.9.7 Nonvolatile memory interface ... 3150
74.9.8 EOUT interface ... 3151
74.9.9 Error signal flow ... 3160
74.10 Register descriptions ... 3160
74.10.1 Control (FCCU_CTRL) ... 3164
74.10.2 Control Key (FCCU_CTRLK) ... 3166
74.10.3 Configuration (FCCU_CFG) ... 3167
74.10.4 Noncritical Fault Configuration (FCCU_NCF_CFG*n*) ... 3171
74.10.5 Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*) ... 3172
74.10.6 Noncritical Fault Status (FCCU_NCF_S*n*) ... 3173
74.10.7 Noncritical Fault Key (FCCU_NCFK) ... 3175

74.10.8 Noncritical Fault Enable (FCCU_NCF_E*n*)....3175
74.10.9 Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*)....3176
74.10.10 Noncritical Fault Timeout (FCCU_NCF_TO)....3177
74.10.11 Configuration-State Timer Interval (FCCU_CFG_TO)....3178
74.10.12 IO Control (FCCU_EINOUT)....3179
74.10.13 Status (FCCU_STAT)....3180
74.10.14 NA Freeze Status (FCCU_N2AF_STATUS)....3182
74.10.15 AF Freeze Status (FCCU_A2FF_STATUS)....3183
74.10.16 NF Freeze Status (FCCU_N2FF_STATUS)....3184
74.10.17 FA Freeze Status (FCCU_F2A_STATUS)....3185
74.10.18 Noncritical Fault Fake (FCCU_NCFF)....3186
74.10.19 IRQ Status (FCCU_IRQ_STAT)....3187
74.10.20 IRQ Enable (FCCU_IRQ_EN)....3189
74.10.21 XTMR (FCCU_XTMR)....3190
74.10.22 Mode Controller Status (FCCU_MCS)....3191
74.10.23 Transient Configuration Lock (FCCU_TRANS_LOCK)....3193
74.10.24 Permanent Configuration Lock (FCCU_PERMNT_LOCK)....3194
74.10.25 Delta T (FCCU_DELTA_T)....3194
74.10.26 IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*)....3195
74.10.27 NMI Enable (FCCU_NMI_EN*n*)....3196
74.10.28 Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*)....3197
74.10.29 Configuration registers....3198
74.10.30 FCCU_CFG register bit value sources (N and C) by event....3199
74.11 Use cases and limitations....3200
74.11.1 Configuration guidelines....3200
74.11.2 Recommendations to configure FCCU....3201

Chapter 75
Self-Test Control Unit (STCU2)

75.1 Chip-specific STCU2 information....3203

75.1.1 STCU configuration....3203
75.1.2 Supported BIST sequence....3204
75.1.3 Device specific STCU Registers....3204
75.1.4 STCU register reset values....3204
75.1.5 Partition overview....3205
75.1.6 STCU On-Line and Off-Line Self-Test sequence preparation....3207
75.1.7 AUTOLOCK_VALUE for register write access and DCF_COMPLETION value for DCF write completion....3207
75.1.8 Wait time for writing to the online registers....3207
75.1.9 On-Line reset generation (only MBIST)....3207
75.2 Introduction....3208
75.3 Main features....3209
75.4 Block diagram....3210
75.5 IPS bus interface....3211
75.6 BISTs and BIST partitions....3213
75.6.1 Definition: BIST....3213
75.6.2 Definition: BIST partition....3213
75.6.3 Example: BIST partitions on a chip....3213
75.6.4 Types of BIST partitions....3213
75.7 BIST sequences....3214
75.7.1 Definition: BIST sequence....3214
75.7.2 Example: Single-phase BIST sequence....3214
75.7.3 Example: Multiphase BIST sequence....3214
75.7.4 Types of BIST sequences....3214
75.7.5 Supported BIST sequences....3215
75.7.6 Definition: Default offline BIST sequence....3215
75.8 Functional description....3215
75.8.1 FSM description....3215
75.8.2 Reset management....3216

75.8.3 Built-in self-test scheduling ... 3216
75.8.4 ABORT management ... 3217
75.8.5 PLL interface ... 3217
75.8.6 FCCU interface ... 3217
75.8.7 Watchdogs ... 3218
75.8.8 SSCM interface ... 3219
75.9 Register description ... 3219
75.9.1 STCU2 Run Register (STCU2_RUN) ... 3223
75.9.2 STCU2 Run Software Register (STCU2_RUNSW) ... 3224
75.9.3 STCU2 SK Code Register (STCU2_SKC) ... 3225
75.9.4 STCU2 Configuration Register (STCU2_CFG) ... 3226
75.9.5 STCU2 Watchdog Granularity (STCU2_WDG) ... 3229
75.9.6 STCU2 Error Register (STCU2_ERR_STAT) ... 3230
75.9.7 STCU2 Error FM Register (STCU2_ERR_FM) ... 3233
75.9.8 STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL) ... 3235
75.9.9 STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM) ... 3240
75.9.10 STCU2 Off-Line MBIST Status High Register (STCU2_MBSH) ... 3242
75.9.11 STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL) ... 3243
75.9.12 STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM) ... 3246
75.9.13 STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH) ... 3248
75.9.14 STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW) ... 3249
75.9.15 STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW) ... 3252
75.9.16 STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW) ... 3254
75.9.17 STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW) ... 3255
75.9.18 STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW) ... 3259
75.9.19 STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW) ... 3261
75.9.20 STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML) ... 3261
75.9.21 STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM) ... 3265
75.9.22 STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH) ... 3267

75.9.23 STCU2 MBIST Control Register (STCU2_MB_CTRL*n*)..... 3268
75.10 Use cases and limitations..... 3269
75.10.1 Offline self-test sequence..... 3269
75.10.2 Online self test sequence..... 3270
75.10.3 Bypass USER Mode..... 3272
75.10.4 Design implementation information..... 3272

Chapter 76
Cyclic Redundancy Check (CRC) Unit

76.1 Introduction..... 3273
76.2 Main features..... 3273
76.2.1 Standard features..... 3273
76.3 Block diagram..... 3274
76.4 External signal description..... 3274
76.4.1 Peripheral bus interface..... 3274
76.5 CRC memory map and registers..... 3275
76.5.1 Configuration Register (CRC_CFG)..... 3276
76.5.2 Input Register (CRC_INP)..... 3277
76.5.3 Current Status Register (CRC_CSTAT)..... 3278
76.5.4 Output Register (CRC_OUTP)..... 3278
76.6 Functional description..... 3279
76.7 Use cases..... 3281
76.7.1 Programming example..... 3281
76.7.2 Register programming..... 3282

Chapter 77
Register Protection

77.1 Chip-specific Register Protection information..... 3285
77.1.1 Register protection configuration..... 3285
77.1.2 Clock Monitor Unit (CMU) protected registers..... 3286
77.1.3 Fast Oscillator Digital Interface (FXOSC) protected registers..... 3286

77.1.4 Clock Generation Module (MC_CGM) protected registers ... 3287
77.1.5 Mode Entry (MC_ME) protected registers ... 3287
77.1.6 Reset Generation Module (MC_RGM) protected registers ... 3288
77.1.7 Memory Error Management Unit (MEMU) protected registers ... 3289
77.1.8 PLL Digital Interface (PLL_DIG) protected registers ... 3289
77.1.9 Power Management Controller digital interface (PMCDIG) protected registers ... 3289
77.1.10 SIUL2 protected registers ... 3290
77.1.11 Slow Oscillator Digital Interface (SXOSC) protected registers ... 3290
77.1.12 LPU Controller (LPU_CTL) protected registers ... 3291
77.2 Overview ... 3291
77.3 Features ... 3292
77.4 Modes of operation ... 3292
77.5 External signal description ... 3293
77.6 Memory map and register definition ... 3293
77.6.1 Memory map ... 3294
77.6.2 Register descriptions ... 3295
77.7 Memory map and registers ... 3295
77.7.1 Soft Lock Bit Register *n* (REG_PROT_SLBRn) ... 3296
77.7.2 Global Configuration Register (REG_PROT_GCR) ... 3298
77.8 Functional description ... 3298
77.8.1 General ... 3299
77.8.2 Change lock settings ... 3299
77.8.3 Access errors ... 3302

Chapter 78
Password and Device Security Module (PASS)

78.1 Chip-specific PASS information ... 3305
78.1.1 Password and Device Security Module (PASS) configuration ... 3305
78.2 Introduction ... 3310
78.2.1 Overview ... 3310

78.2.2 Features ... 3311
78.2.3 Modes of operation ... 3311
78.3 Memory map and register definition ... 3311
78.3.1 Life Cycle Status Register (PASS_LCSTAT) ... 3313
78.3.2 Challenge Selector Register (PASS_CHSEL) ... 3315
78.3.3 Challenge Status Register (PASS_CSTAT) ... 3315
78.3.4 Challenge Input Register (PASS_CIN*n*) ... 3316
78.3.5 Password Group *n* - Lock 0 Status Register (PASS_LOCK0_PG*n*) ... 3316
78.3.6 Password Group *n* - Lock 1 Status Register (PASS_LOCK1_PG*n*) ... 3318
78.3.7 Password Group *n* - Lock 2 Status Register (PASS_LOCK2_PG*n*) ... 3319
78.3.8 Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*) ... 3320
78.4 DCF clients ... 3324
78.4.1 Censorship ... 3324
78.4.2 Test mode enable DCF Client ... 3325
78.5 Functional description ... 3326
78.5.1 Censorship ... 3326
78.5.2 Debug Interface Access ... 3326
78.5.3 Flash Memory Read Protection ... 3327
78.5.4 Flash Memory Write and Erase Protection ... 3328
78.5.5 Test mode enable ... 3328
78.5.6 Test mode enable password ... 3329
78.6 Initialization/application information ... 3330
78.6.1 Reset ... 3330
78.6.2 Setting lock bits in a password group ... 3330

Chapter 79
Tamper Detect Module (TDM)

79.1 Chip-specific TDM information ... 3333
79.1.1 Tamper Detection Module (TDM) configuration ... 3333
79.1.2 Diary Base Address (DBA) DCF client ... 3333

79.1.3 Tamper Region Override (TO) DCF client....3334
79.1.4 OTPENn DCF clients....3334
79.1.5 TDRx_LOCKn DCF clients....3334
79.1.6 TDM register reset value....3335
79.2 Overview....3335
79.3 Features....3337
79.4 External signal description....3337
79.5 DCF clients....3337
79.5.1 Diary Base Address (DBA) DCF client....3338
79.5.2 Tamper Region Override (TO) DCF client....3339
79.5.3 Software Tamper Region Override Disable (STO_DIS) DCF client....3339
79.5.4 One Time Programmable Enable (OTPEN0) DCF client....3340
79.5.5 One Time Programmable Enable (OTPEN1) DCF client....3341
79.5.6 One Time Programmable Enable (OTPEN2) DCF client....3342
79.5.7 One Time Programmable Enable (OTPEN3) DCF client....3343
79.5.8 Tamper Region Assignment DCF client (TDRn_LOCK0)....3343
79.5.9 Tamper Region Assignment DCF client (TDRn_LOCK1)....3344
79.5.10 Tamper Region Assignment DCF client (TDRn_LOCK2)....3345
79.5.11 Tamper Region Assignment DCF client (TDRn_LOCK3)....3346
79.6 Memory Map/Register Definition....3347
79.6.1 TDR Status Register (TDM_TDRSR)....3347
79.6.2 Last Flash Programmed Address Register (TDM_LFPAR)....3350
79.6.3 Diary Base Address (TDM_DBA)....3350
79.6.4 Software Tamper Override Key Region (TDM_STO_KEY*n*)....3351
79.7 Functional description....3351
79.7.1 Flash erase counter and tamper detection....3351
79.7.2 Assigning blocks to Tamper Detect Regions (TDR)....3353
79.7.3 Diary....3354
79.7.4 Specifying the diary base address....3354

79.7.5 TDR lock status....3355
79.7.6 TDR override....3356
79.7.7 One time programmable (OTP)....3356

Chapter 80
Flash Memory Programming and Configuration

80.1 Introduction....3359
80.2 Selection of flash memory blocks for erase....3361
80.3 Non-secure write protection....3364
80.4 Secure write protection....3366
80.4.1 Implementing secure write protection....3367
80.4.2 Overriding secure write protection....3371
80.5 Secure read protection....3373
80.5.1 Implementing secure read protection....3375
80.5.2 Overriding secure read protection....3379
80.6 Debug port enable/disable....3380
80.7 Tamper detection....3380
80.7.1 Implementing tamper detection....3380
80.7.2 Creating the tamper detect diary....3381
80.7.3 Assigning blocks to Tamper Detection Regions (TDRs)....3383
80.7.4 Overriding tamper detection....3384
80.8 Implementing OTP....3385
80.9 Implementing test mode disable....3385
80.9.1 Unconditional test mode disable seal....3386
80.9.2 Passcode-protected test mode disable seal....3387
80.9.3 Selecting flash memory blocks for test mode disable seal....3387
80.10 Security configuration planning....3391
80.10.1 Hardware Security Module (HSM)....3391
80.10.2 Creating password groups....3392
80.10.3 Planning secure write protection....3392

80.10.4 Planning secure read protection..... 3393
80.10.5 Planning debug port enable/disable..... 3393
80.10.6 Planning OTP flash memory block assignment..... 3393
80.10.7 Planning factory test mode disable..... 3393

Chapter 1
About This Manual

1.1 Audience

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

1.2 Organization

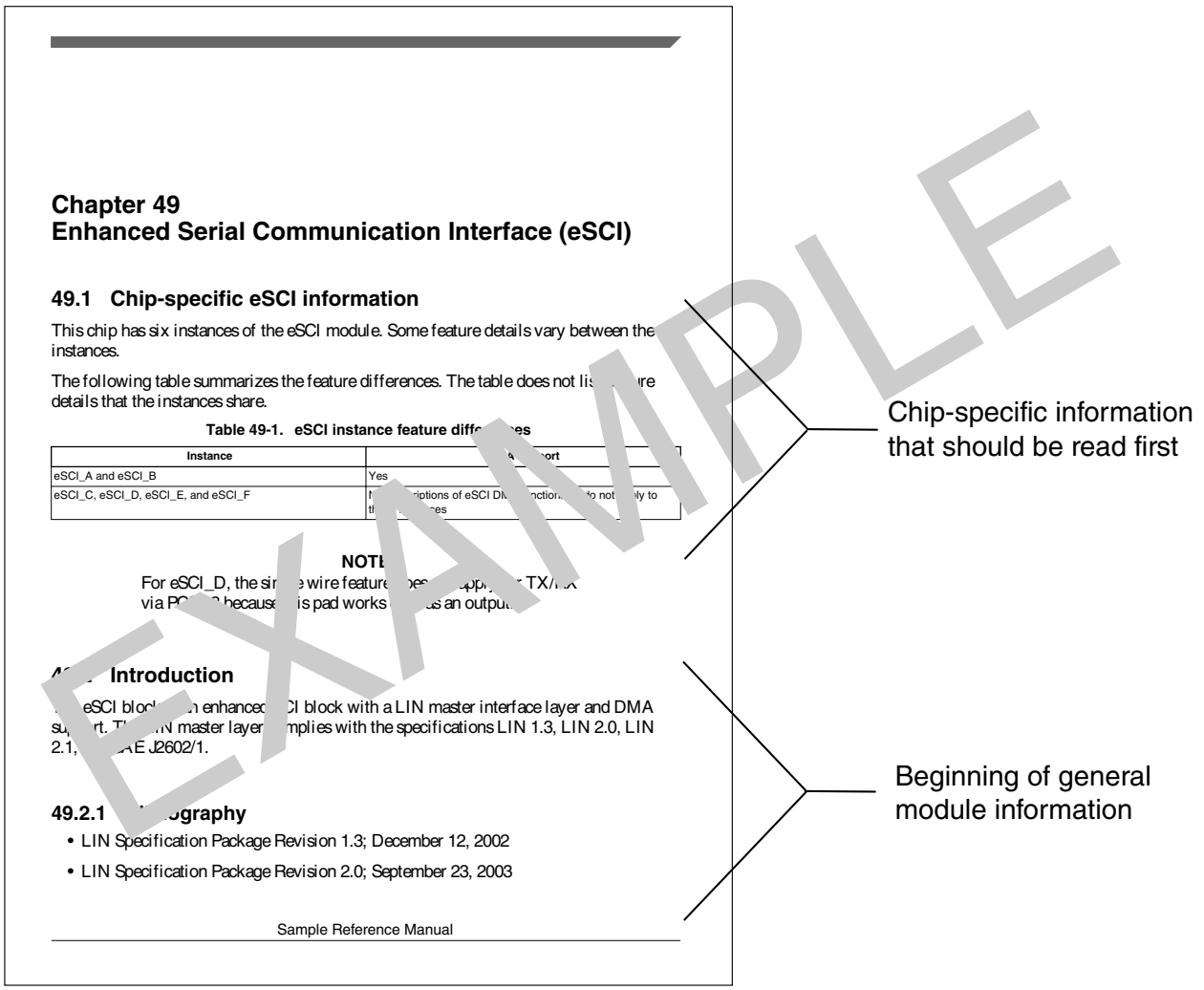
This manual has two main sets of chapters.

1. Chapters in the first set contain information that applies to all components on the chip.
2. Chapters in the second set are organized into functional groupings that detail particular areas of functionality.
   - Examples of these groupings are clocking, timers, and communication interfaces.
   - Each grouping includes chapters that provide a technical description of individual modules.

1.3 Module descriptions

Each module chapter has two main parts:

- The first section, *Chip-specific [module name] information*, provides details such as the number of module instances on the chip and connections between the module and other modules. Read this section *first* because its content is crucial to understanding the information in other sections of the chapter.
- The subsequent sections provide general information about the module, including its signals, registers, and functional description.

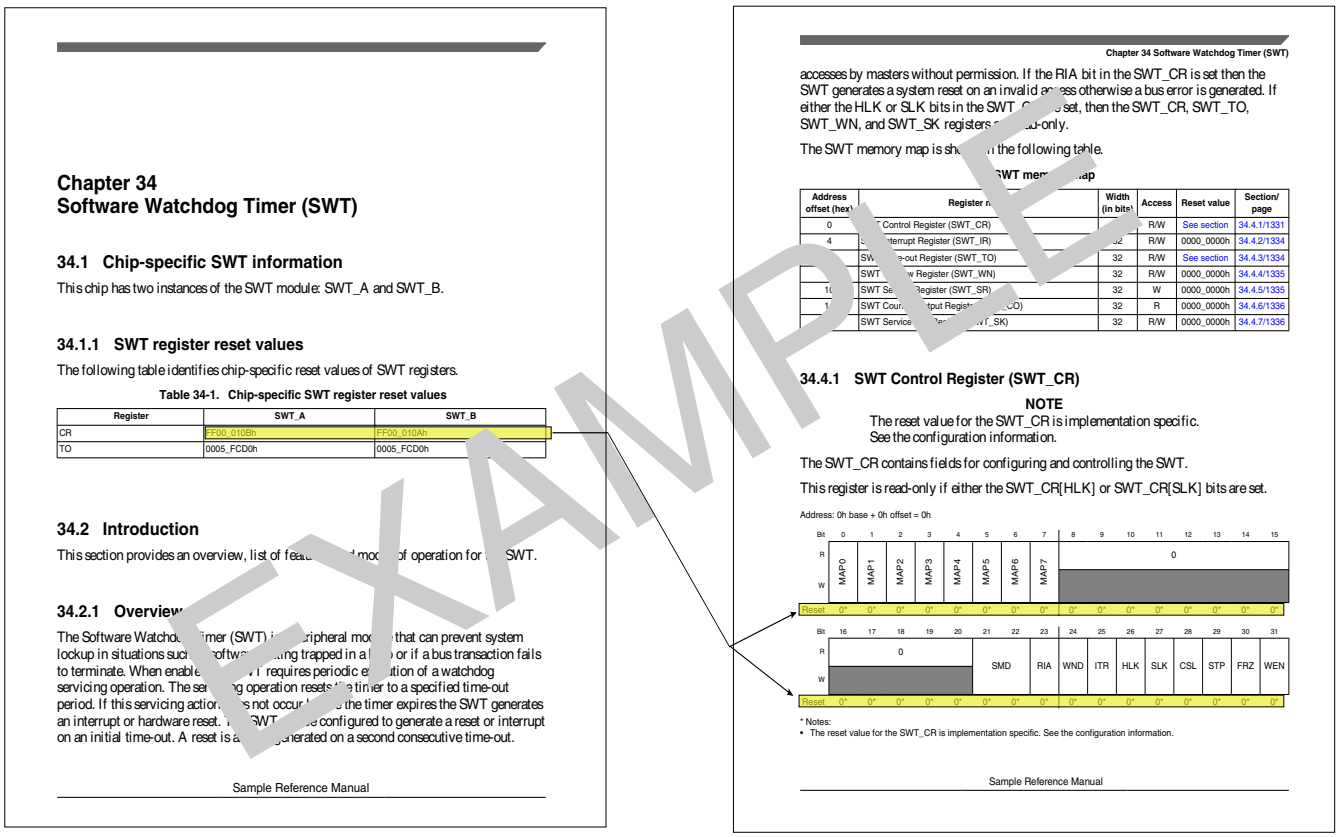


EXAMPLE

Figure 1-1. Example: chapter chip-specific information and general module information

1.3.1 Example: chip-specific information that clarifies content in the same chapter

The example below shows chip-specific information that clarifies general module information presented later in the chapter. In this case, the chip-specific register reset values supercede the reset values that appear in the register diagram.



Chapter 34
Software Watchdog Timer (SWT)

34.1 Chip-specific SWT information

This chip has two instances of the SWT module: SWT_A and SWT_B.

34.1.1 SWT register reset values

The following table identifies chip-specific reset values of SWT registers.

Table 34-1. Chip-specific SWT register reset values

Register	SWT_A	SWT_B
CR	FF00_010Bh	FF00_010Ah
TO	0005_FCD0h	0005_FCD0h

34.2 Introduction

This section provides an overview, list of feat... mod... of operation for ... SWT.

34.2.1 Overview

The Software Watchdo... imer (SWT) i... ripheral mo... that can prevent system lockup in situations suc... softw... ting trapped in a l... or if a bus transaction fails to terminate. When enable... T requires periodic e... ution of a watchdog servicing operation. The ser... g operation resets ... timer to a specified time-out period. If this servicing actio... s not occur ... the timer expires the SWT generates an interrupt or hardware reset. ... SWT ... be configured to generate a reset or interrupt on an initial time-out. A reset is a... generated on a second consecutive time-out.

Sample Reference Manual

Chapter 34 Software Watchdog Timer (SWT)

accesses by masters without permission. If the RIA bit in the SWT_CR is set then the SWT generates a system reset on an invalid a... ss otherwise a bus error is generated. If either the HLK or SLK bits in the SWT_C... set, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers ... -only.

The SWT memory map is sh... n the following table.

SWT mem... ap

Address offset (hex)	Register n...	Width (in bits)	Access	Reset value	Section/ page
0	... Control Register (SWT_CR)		R/W	See section	34.4.1/1331
4	... terrupt Register (SWT_IR)	32	R/W	0000_0000h	34.4.2/1334
	SW... e-out Register (SWT_TO)	32	R/W	See section	34.4.3/1334
	SWT ... w Register (SWT_WN)	32	R/W	0000_0000h	34.4.4/1335
10	SWT Se... Register (SWT_SR)	32	W	0000_0000h	34.4.5/1335
1	SWT Cour... tput Regist... CO)	32	R	0000_0000h	34.4.6/1336
	SWT Service ... (SWT_SK)	32	R/W	0000_0000h	34.4.7/1336

34.4.1 SWT Control Register (SWT_CR)

NOTE
The reset value for the SWT_CR is implementation specific.
See the configuration information.

The SWT_CR contains fields for configuring and controlling the SWT.

This register is read-only if either the SWT_CR[HLK] or SWT_CR[SLK] bits are set.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0							
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SMD	RIA	WND	ITR	HLK	SLK	CSL	STP	FRZ	WEN	
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value for the SWT_CR is implementation specific. See the configuration information.

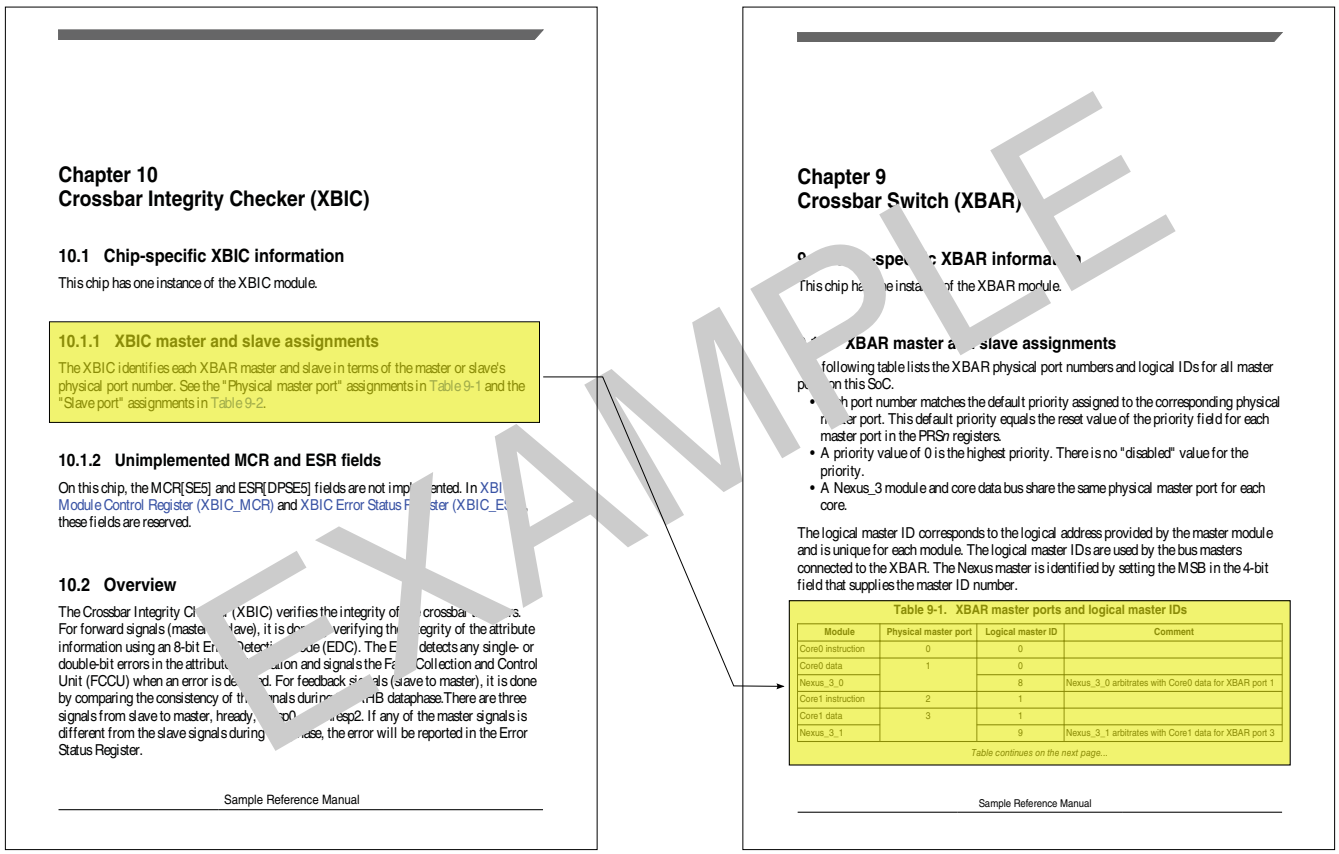
Sample Reference Manual

EXAMPLE

Figure 1-2. Example: chip-specific information that clarifies content in the same chapter

1.3.2 Example: chip-specific information that refers to a different chapter

The chip-specific information below refers to another chapter's chip-specific information. In this case, read both sets of chip-specific information before reading further in the chapter.



Chapter 10
Crossbar Integrity Checker (XBIC)

10.1 Chip-specific XBIC information

This chip has one instance of the XBIC module.

10.1.1 XBIC master and slave assignments

The XBIC identifies each XBAR master and slave in terms of the master or slave's physical port number. See the "Physical master port" assignments in Table 9-1 and the "Slave port" assignments in Table 9-2.

10.1.2 Unimplemented MCR and ESR fields

On this chip, the MCR[SE5] and ESR[DPSE5] fields are not implemented. In XBIC Module Control Register (XBIC_MCR) and XBIC Error Status Register (XBIC_ES... these fields are reserved.

10.2 Overview

The Crossbar Integrity Ch... (XBIC) verifies the integrity of ... crossbar ... For forward signals (maste... lave), it is do... verifying th... egrity of the attribute information using an 8-bit Er... Detecti... ue (EDC). The E... detects any single- or double-bit errors in the attribut... ation and signals the Fa... Collection and Control Unit (FCCU) when an error is de... d. For feedback si... ls (slave to master), it is done by comparing the consistency of th... nals during ... hB dataphase.There are three signals from slave to master, hready, ... so0, ... esp2. If any of the master signals is different from the slave signals during ... ase, the error will be reported in the Error Status Register.

Sample Reference Manual

Chapter 9
Crossbar Switch (XBAR)

9 ... -spe... : XBAR informa...

This chip ha... e insta... of the XBAR module.

... XBAR master a... slave assignments

... following table lists the XBAR physical port numbers and logical IDs for all master p... on this SoC.

- ... h port number matches the default priority assigned to the corresponding physical m... er port. This default priority equals the reset value of the priority field for each master port in the PRSn registers.
- A priority value of 0 is the highest priority. There is no "disabled" value for the priority.
- A Nexus_3 module and core data bus share the same physical master port for each core.

The logical master ID corresponds to the logical address provided by the master module and is unique for each module. The logical master IDs are used by the bus masters connected to the XBAR. The Nexus master is identified by setting the MSB in the 4-bit field that supplies the master ID number.

Table 9-1. XBAR master ports and logical master IDs

Module	Physical master port	Logical master ID	Comment
Core0 instruction	0	0	
Core0 data	1	0	
Nexus_3_0		8	Nexus_3_0 arbitrates with Core0 data for XBAR port 1
Core1 instruction	2	1	
Core1 data	3	1	
Nexus_3_1		9	Nexus_3_1 arbitrates with Core1 data for XBAR port 3

Table continues on the next page...

Sample Reference Manual

EXAMPLE

Figure 1-3. Example: chip-specific information that refers to a different chapter

1.4 Register descriptions

Module chapters present register information in:

- Memory maps containing:
  - An offset from the module's base address
  - The name and acronym/abbreviation of each register
  - The width of each register (in bits)
  - Each register's reset value
  - The page number on which each register is described
- Register figures
- Field-description tables
- Associated text

The register figures show the field structure using the conventions in the following figure.

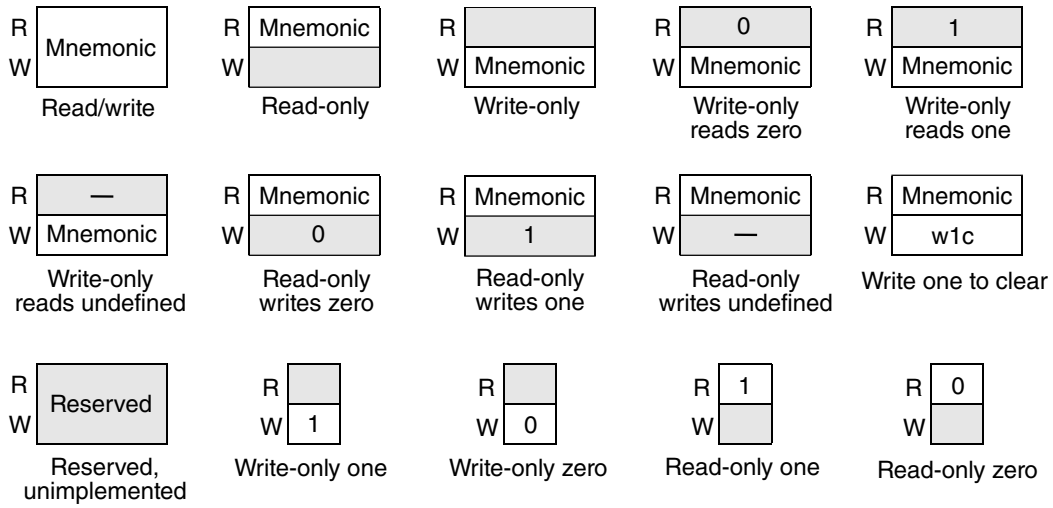


Figure 1-4. Register figure conventions

1.5 Conventions

1.5.1 Notes, Cautions, and Warnings

Note, Caution, and Warning notices appear throughout this manual. Each notice type alerts readers to a specific kind of information.

NOTE

Notes convey information that may be tangential to a topic or that may not apply to all readers.

CAUTION

Caution notices call out information that readers should know before taking further action. Cautions frequently point to trouble spots that may damage the chip or board.

WARNING

Warning notices inform readers about actions that could result in unwanted consequences, especially those that may cause bodily injury.

1.5.2 Numbering systems

The following suffixes identify different numbering systems:

This suffix	Identifies a
b	Binary number. For example, the binary equivalent of the number 5 is written 101b. In some cases, binary numbers are shown with the prefix *0b*.
d	Decimal number. Decimal numbers are followed by this suffix only when the possibility of confusion exists. In general, decimal numbers are shown without a suffix.
h	Hexadecimal number. For example, the hexadecimal equivalent of the number 60 is written 3Ch. In some cases, hexadecimal numbers are shown with the prefix *0x*.

1.5.3 Typographic notation

The following typographic notation is used throughout this document:

Example	Description
placeholder, x	Items in italics are placeholders for information that you provide. Italicized text is also used for the titles of publications and for emphasis. Plain lowercase letters are also used as placeholders for single letters and numbers.
`code`	Fixed-width type indicates text that must be typed exactly as shown. It is used for instruction mnemonics, directives, symbols, subcommands, parameters, and operators. Fixed-width type is also used for example code. Instruction mnemonics and directives in text and tables are shown in all caps; for example, BSR.
SR[SCM]	A mnemonic in brackets represents a named field in a register. This example refers to the Scaling Mode (SCM) field in the Status Register (SR).
REVNO[6:4], XAD[7:0]	Numbers in brackets and separated by a colon represent either: • A subset of a register's named field For example, REVNO[6:4] refers to bits 6–4 that are part of the COREREV field that occupies bits 6–0 of the REVNO register. • A continuous range of individual signals of a bus For example, XAD[7:0] refers to signals 7–0 of the XAD bus.

1.5.4 Special terms

The following terms have special meanings:

Term	Meaning
asserted	Refers to the state of a signal as follows: • An active-high signal is asserted when high (1). • An active-low signal is asserted when low (0).
deasserted	Refers to the state of a signal as follows: • An active-high signal is deasserted when low (0). • An active-low signal is deasserted when high (1). In some cases, deasserted signals are described as *negated*.

Table continues on the next page...

Term	Meaning
reserved	Refers to a memory space, register, field, or programming setting. Writes to a reserved location can result in unpredictable functionality or behavior. • Do not modify the default value of a reserved programming setting, such as the reset value of a reserved register field. • Consider undefined locations in memory to be reserved.
w1c	Write 1 to clear: Refers to a register bitfield that must be written as 1 to be "cleared."

Chapter 2
Introduction

2.1 Introduction

The MPC5746C is a 32-bit Power Architecture based microcontroller unit (MCU) for automotive applications and builds on the successful legacy of the MPC56xx family of microcontrollers.

There are two application processor cores on this device.

Table 2-1. Processor cores

Computational Shell		
Core0	Main Core 0	e200z420n3
Core1	Peripheral Core	e200z210n3

2.1.1 Target Applications

The family of devices are designed to address a wide variety of automotive applications including but not limited to the door modules, seat modules, central body, vehicle body controllers, smart junction box, front module applications, high end gateway or combined body controller and gateway applications.

2.2 Block diagram

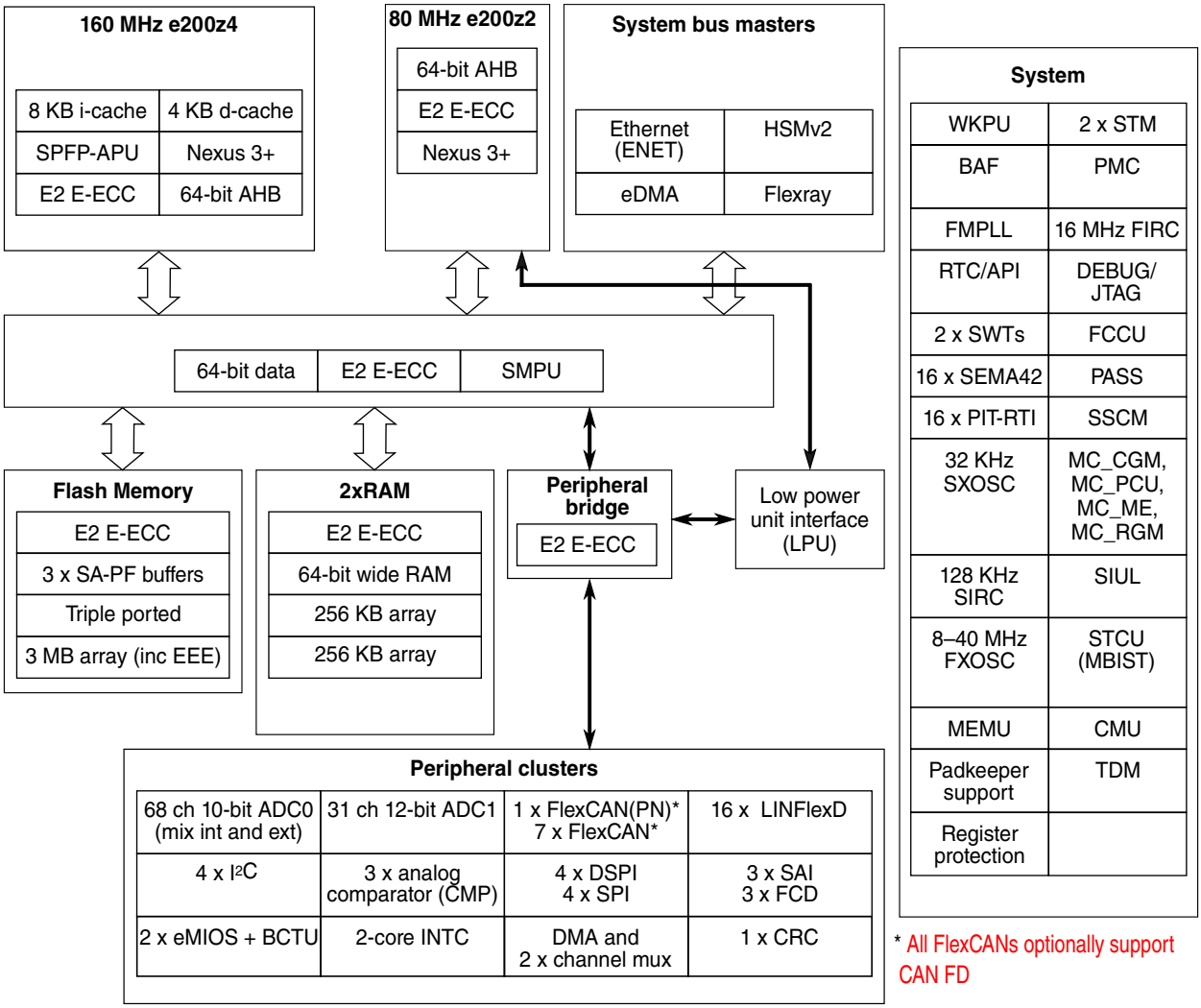


Figure 2-1. MPC5746C block diagram

2.3 Family comparison

The following table provides a summary of the different members of the MPC5746C family and their proposed features. This information is intended to provide an understanding of the range of functionality offered by this family. For full details of all of the family derivatives please contact your marketing representative.

NOTE

All optional features (Flash memory, RAM, Peripherals) start with lowest number or address (e.g., FlexCAN0) and end at highest available number or address (e.g., MPC574xB/C have 6 CAN, ending with FlexCAN5).

Table 2-2. MPC5746C Family Comparison[1]

Feature	MPC5745B	MPC5744B	MPC5746B	MPC5744C	MPC5745C	MPC5746C
CPUs	e200z4	e200z4	e200z4	e200z4 e200z2	e200z4 e200z2	e200z4 e200z2
FPU	e200z4	e200z4	e200z4	e200z4	e200z4	e200z4
Maximum Operating Frequency[2]	160MHz (Z4)	160MHz (Z4)	160MHz (Z4)	160MHz (Z4) 80MHz (Z2)	160MHz (Z4) 80MHz (Z2)	160MHz (Z4) 80MHz (Z2)
Flash memory	2 MB	1.5 MB	3 MB	1.5 MB	2 MB	3 MB
EEPROM support	Emulated up to 64K			Emulated up to 64K		
RAM	256 KB	192 KB	384 KB (Optional 512KB)[3]	192 KB	256 KB	384 KB (Optional 512KB)[3]
ECC	End to End					
SMPU	16 entry					
DMA	32 channels					
10-bit ADC	36 Standard channels 32 External channels					
12-bit ADC	15 Precision channels 16 Standard channels					
Analog Comparator	3					
BCTU	1					
SWT	1, SWT[0]			2[4]		
STM	1, STM[0]			2		
PIT-RTI	16 channels PIT 1 channels RTI					
RTC/API	1					
Total Timer I/O[5]	64 channels 16-bits					
LINFlexD	1 Master and Slave (LINFlexD[0], 11 Master (LINFlexD[1:11])			1 Master and Slave (LINFlexD[0], 15 Master (LINFlexD[1:15])		
FlexCAN	6 with optional CAN FD support (FlexCAN[0:5])			8 with optional CAN FD support (FlexCAN[0:7])		
DSPI/SPI	4 x DSPI 4 x SPI					
I^2C	4	4	4	4		
SAI/I^2S	3	3	3	3		
FXOSC	8 - 40 MHz					
SXOSC	32 KHz					
FIRC	16 MHz					
SIRC	128 KHz					

Table continues on the next page...

Table 2-2. MPC5746C Family Comparison[1] (continued)

Feature	MPC5745B	MPC5744B	MPC5746B	MPC5744C	MPC5745C	MPC5746C
FMPLL	1					
Low Power Unit (LPU)	Yes					
FlexRay 2.1 (dual channel)	Yes, 128 MB	Yes, 128 MB	Yes, 128 MB	Yes, 128 MB		
Ethernet (RMII, MII + 1588, Muti queue AVB support)	1	1	1	1		
CRC	1					
MEMU	2					
STCU2	1					
HSM-v2 (security)	Optional					
Censorship	Yes					
FCCU	1					
Safety level	Specific functions ASIL-B certifiable					
User MBIST	Yes					
I/O Retention in Standby	Yes					
GPIO[6]	Up to 264 GPI and up to 246 GPIO					
Debug	JTAGC, cJTAG					
Nexus	Z4 N3+ (Only available on 324BGA (development only)) Z2 N3+ (Only available on 324BGA (development only))					
Packages	176 LQFP-EP 256 BGA 100 BGA	176 LQFP-EP 256 BGA 100 BGA	176 LQFP-EP 256 BGA 100 BGA	176 LQFP-EP 256 BGA 100 BGA	176 LQFP-EP 256 BGA 100 BGA	176 LQFP-EP 256 BGA, 324 BGA (development only) 100 BGA

1. Feature set dependent on selected peripheral multiplexing, table shows example. Peripheral availability is package dependent.
2. Based on 125°C ambient operating temperature and subject to full device characterization.
3. Contact NXP representative for part number
4. Additional SWT included when HSM option selected
5. See device datasheet and reference manual for information on to timer channel configuration and functions.
6. Estimated I/O count for largest proposed packages based on multiplexing with peripherals.

Table 2-3. MPC5746C Family Comparison - NVM Memory Map 1

Start Address	End Address	Flash block	RWW partition	MPC5744	MPC5745	MPC5746
0x01000000	0x0103FFFF	256 KB code Flash block 0	6	available	available	available

Table continues on the next page...

Table 2-3. MPC5746C Family Comparison - NVM Memory Map 1 (continued)

Start Address	End Address	Flash block	RWW partition	MPC5744	MPC5745	MPC5746
0x01040000	0x0107FFFF	256 KB code Flash block 1	6	available	available	available
0x01080000	0x010BFFFF	256 KB code Flash block 2	6	available	available	available
0x010C0000	0x010FFFFF	256 KB code Flash block3	6	available	available	available
0x01100000	0x0113FFFF	256 KB code Flash block 4	6	not available	available	available
0x01140000	0x0117FFFF	256 KB code Flash block 5	7	not available	available	available
0x01180000	0x011BFFFF	256 KB code Flash block 6	7	not available	not available	available
0x011C0000	0x011FFFFF	256 KB code Flash block 7	7	not available	not available	available
0x01200000	0x0123FFFF	256 KB code Flash block 8	7	not available	not available	available
0x01240000	0x0127FFFF	256 KB code Flash block 9	7	not available	not available	not available

Table 2-4. MPC5746C Family Comparison - NVM Memory Map 2

Start Address	End Address	Flash block	RWW partition	MPC5744B MPC5745B MPC5746B	MPC5744C MPC5745C MPC5746C
0x00F90000	0x00F93FFF	16 KB data Flash	2	available	available
0x00F94000	0x00F97FFF	16 KB data Flash	2	available	available
0x00F98000	0x00F9BFFF	16 KB data Flash	2	available	available
0x00F9C000	0x00F9FFFF	16 KB data Flash	2	available	available
0x00FA0000	0x00FA3FFF	16 KB data Flash	3	not available	available
0x00FA4000	0x00FA7FFF	16 KB data Flash	3	not available	available
0x00FA8000	0x00FABFFF	16 KB data Flash	3	not available	available
0x00FAC000	0x00FAFFFF	16 KB data Flash	3	not available	available

Table 2-5. MPC5746C Family Comparison - RAM Memory Map

Start Address	End Address	Allocated size	Description	MPC5744	MPC5745	MPC5746
0x40000000	0x40001FFF	8 KB	SRAM0	available	available	available
0x40002000	0x4000FFFF	56 KB	SRAM1	available	available	available
0x40010000	0x4001FFFF	64 KB	SRAM2	available	available	available
0x40020000	0x4002FFFF	64 KB	SRAM3	available	available	available
0x40030000	0x4003FFFF	64 KB	SRAM4	not available	available	available
0x40040000	0x4004FFFF	64 KB	SRAM5	not available	not available	available

Table continues on the next page...

Table 2-5. MPC5746C Family Comparison - RAM Memory Map (continued)

Start Address	End Address	Allocated size	Description	MPC5744	MPC5745	MPC5746
0x40050000	0x4005FFFF	64 KB	SRAM6	not available	not available	available
0x40060000	0x4006FFFF	64 KB	SRAM7	not available	not available	optional
0x40070000	0x4007FFFF	64 KB	SRAM8	not available	not available	optional

2.4 Feature list

On-chip modules available within the device include the following features:

Table 2-6. Feature List

Cores	• 1 x 160 MHz Power Architecture® e200Z4 Dual issue, 32-bit CPU
	  • Single precision floating point operations
	  • 8 KB instruction cache and 4 KB data cache
	  • Only variable length encoding (VLE) for significant code density improvements
	• 1 x 80 MHz Power Architecture® e200Z2 Single issue, 32-bit CPU
	  • Using variable length encoding (VLE) for significant code size footprint reduction
	• End-to-End ECC ensuring full protection of all data accesses throughout the system, from each of the core through the crossbar and into the memories and peripheral bridge

Supported memories	• 3 MB on-chip flash supported with the flash controller • 3 x flash page buffers (3 port flash controller) to improve access time to code and data held in flash • 512 KB on-chip SRAM across two RAM ports
Clocks	• 8-40 MHz Fast External Crystal Oscillator (FXOSC) • 16 MHz Fast Internal RC Oscillator (FIRC) • 128 KHz Slow Internal RC Oscillator(SIRC) • 32 KHz Slow External Crystal Oscillator (SXOSC) • Clock Monitor Unit (CMU) to support monitoring of critical clocks • Frequency modulated phase-locked loop (FMPLL) • Sophisticated clocking architecture allowing core and peripheral clocks to be scaled independently of the peripheral operation (some restrictions may apply)
System	• Two System Memory Protection Unit (SMPU) with up to 32 region descriptors and 16-byte region granularity • 16 Semaphores to manage access to shared resource • Interrupt controller (INTC) capable of routing interrupts to any CPU • Single crossbar switch architecture for concurrent access to peripherals, flash, and RAM from multiple bus masters • 32-channels eDMA controller with multiple transfer request sources using DMAMUX • Body Cross Trigger Unit (BCTU) to enable synchronization of ADC conversions with a timer event from the eMIOS or from the PIT • Capable of triggering groups of channels • Supports wake-up from low power modes via the wakeup unit (WKPU)
Communications	• Four Deserial Peripheral Interface (DSPI) • Four Serial Peripheral interface (SPI) • 16 serial communication interface (LIN) • Eight enhanced FlexCAN modules with configurable buffers • Each instance supports DMA on RxFIFO

Table continues on the next page...

Table 2-6. Feature List (continued)

	• Low power mode with Pretended Networking support on a single FlexCAN (FlexCAN0) • CAN FD support • Four inter-IC communication interface (I2C) • One dual-channel FlexRay Controller with 128 Message buffers and stopwatch function for synchronisation to OS • One ENET complex (10/100 Ethernet) that supports 1588 and MII/RMII • Enhanced ENET complex supporting multiple queues • Mutiple queue AVB support • Three Synchronous Audio Interface (SAI) capable of supporting six stereo audio channels • Three Fractional clock dividers (FCD) operating in conjunction with the SAIs
Analog	• Two analog-to-digital converters (ADC), one 10-bit and one 12-bit • 12-bit ADC has independent reference on larger packages • Each ADC can be independently 3.3 V or 5 V • Three analogue comparators (CMP) providing up to 24 compare inputs • Supports analog input comparison without any software supervision during STANDBY mode • Supports four instances of IGF, one for each HF interface
Timers	• Real Time Counter (RTC) with clock source from internal 128 KHz SIRC, 32 KHz SXOSC, 16 MHz FIRC or the 8-40 MHz FXOSC • Two System Timer Module (STM), each with 4 compare registers. • Two Software WatchDog Timers (SWT) clocked by the SIRC, one additional in HSM • 16 Periodic Interrupt Timers (PIT) channels with 32-bit counter resolution • Channels can be concatenated together to produce a 64-bit resolution • RTI also supported and clocked by FXOSC • 64 Configurable Enhanced Modular Input Output Subsystem (eMIOS) channels • Provides a range of 16-bit input capture, output compare, and pulse width modulation functions • Improved flexibility by addition of second global counter bus and second global pre-scaler
Boot support	• Boot Assist Flash (BAF) supports internal flash programming via a serial link (LIN/SCI and FlexCAN) • BAF runs initialization routines at the device boot • Device always boot on e200Z4 • Device can boot from any core on exit from STANDBY or LPU_RUN
Input output pin configuration	• Maximum amount of general purpose pins supporting input and output operations (Number supported is package dependent) • General purpose I/O support state retention in all low power modes. State retention controlled by pad keeper logic. Limted I/O supports state retention in Power Domain 0 (PD0) and Power Domain 1 (PD1) • Configurable I/O domains supporting Ethernet and general I/O (package dependent)
Power management	• On-chip power management controller (PMC-Dig) • External regulation mode where low voltage input supply of ~1.25 V (+/- 4 %) is regulated and supplied through on board power supply • Internal regulation mode where low voltage input supply of ~1.25 V (+/- 4 %) is regulated by on-chip PMC and supplied via external ballast transistor • Internal regulation mode where low voltage input supply of ~1.25 V (+/- 4 %) is regulated by on-chip PMC and supplied via internal ballast transistor
Debug	• e200Z2 core: Nexus development interface (NDI) per IEEE-ISTO 5001-2012 Class 3+ • Only class 1 debug interface supported during LPU mode • e200Z4 core: Nexus development interface (NDI) per IEEE-ISTO 5001-2012 Class 3+

Table continues on the next page...

Table 2-6. Feature List (continued)

	• Device/board boundary Scan testing supported with per JTAG (IEEE 1149.1) and cJTAG (1149.7) • Possible to trace SRAM accesses via nexus auxiliary interface
Safety	• One hardware Cyclic Redundancy Check (CRC) module • Two Memory Error Management Unit (MEMU) • One Fault Collection and Control Unit (FCCU) • One Self-Test Control Unit (STCU) • Targeted for specific device functions being ASIL-B certifiable
Security	• One Password and Device Security Module (PASS) implementing password-based read and write protection for flash blocks • Tamper Detection Module (TDM) provides flash memory write protection mechanism • Register protection supported on key registers • Hardware Security Module (HSM) enabling advanced security management

2.5 Package Options

The following package options are available in the device:

- 100-pin BGA, 1mm ball pitch, 11mm x 11mm outline
- 176-pin LQFP, 0.5mm pitch, 24mm x 24mm outline
- 256-pin BGA, 1mm ball pitch, 17mm x 17mm outline
- 324-pin BGA, 1mm ball pitch, 19mm x 19mm outline (development only)

2.6 Modes of Operation

This family of devices will have two types of modes, SYSTEM modes and USER modes.

SYSTEM Modes

SYSTEM modes are entered automatically on system events or on application request. There are three SYSTEM modes: RESET, SAFE, and DRUN as described in the table below.

Table 2-7. System Modes

Mode	Description	Entry	Exit
RESET	System remains in this mode until all resources are available for S/W to take device control. This mode manages hardware	System reset assertion from RGM	System reset deassertion from RGM

Table continues on the next page...

Table 2-7. System Modes (continued)

Mode	Description	Entry	Exit
	initialization of flash and various chip configuration registers		
SAFE	May be entered on detection of an error, reaction of which has been pre-configured via MC_RGM and FCCU. Forces system into pre-defined safe configuration such as the system clock source switches to internal fast RC oscillator and output pins are forced into high impedance (Hi-z) from which the system may try to recover.	Hardware fail, Software request from DRUN, RUNx	System reset assertion, DRUN via Software
DRUN	Default entry mode for the embedded software. Provides full system accessibility & enables system configuration. BAF is executed in DRUN mode if exiting from RESET mode or STANDBY mode provided its is configured to execute BAF on standby exit	Reset removed, from any of the mode SAFE and RUNx via software configuration, from LPU_RUN via software configuration, STANDBY/ LPU_STANDBY on wakeup	It is possible to transition to all the modes from DRUN

USER Modes

USER modes are used to control the device performance and consumption during normal application operation. The modes are organized according to the performance provided by the device in each mode. There are various USER modes as described in the table below, depending on the specific device: RUNx, STOP, STANDBY, LPU_RUN, LPU_STOP and LPU_STANDBY.

Table 2-8. User Modes

Mode	Description	Entry	Exit
RUN0, RUN1, RUN2, RUN3	These are software running modes where most processing activity is done. These various run modes allow to enable different clock & power configurations of the system with respect to each other	software request from DRUN, interrupt event from wakeup request from STOP	Device can enter in all modes SAFE, RUNx, STOP, RESET, LPU_RUN via software
STOP0	Advanced low-power modes during which the clock to the core is disabled. STOP0 may be configured to switch off most of the peripherals including oscillator for efficient	Software request from RUNx	Reset asserted, SAFE on hardware failure, RUNx on wakeup event

Table continues on the next page...

Table 2-8. User Modes (continued)

Mode	Description	Entry	Exit
	power management at the cost of higher wakeup latency		
STANDBY/LPU_STANDBY	Reduced-leakage low-power mode during which power is cut off from most of device and content is lost and must be restored from backup	Software request from RUNx, DRUN, LPU_RUN modes	Device will enter in DRUN mode on STANDBY exit and LPU_RUN on LPU_STANDBY exit. It is possible to exit from LPU_STANDBY to DRUN if LPU_MCR[DIRECT_DRUN] is configured.

Advanced Low Power USER modes: Advanced Low Power USER modes are used to control the device performance and consumption during advanced low power operation. The modes are organized according to the performance provided by the device in each mode.

There are three modes on this device: LPU_RUN, LPU_STOP, and LPU_SLEEP.

Table 2-9. Advanced Low Power USER modes

Mode	Description	Entry	Exit
LPU_RUN	In LPU_RUN mode only small subsystem with Z2 core and some communication peripheral is powered. Other cores and majority of modules are power gated. Only FIRC and FXOSC can be source of system clock as PLL is power gated.	Software request from RUNx, DRUN, from LPU_STANDBY and LPU_STOP on wakeup	Device will enter into DRUN mode on LPU_RUN exit
LPU_STOP	In LPU_STOP mode, Z2 core enters the low-power state and its clock is gated. FlexCAN pretended network feature is available but all peripherals are stopped, clocks are gated to all modules in LPU domain. LPU_STOP may be configured to switch off oscillators for efficient power management at the cost of higher wakeup latency	On software request from LPU_RUN	Device will enter into LPU_RUN mode on interrupt from FlexCAN, WKPU or SWT timeout
LPU_SLEEP	Z2 core is stopped and clock gated by hardware. LPU_SLEEP may be configured to clock gated module clocks by configuring LPU_MDIS	On software request from LPU_RUN	The device can exit to LPU_RUN mode on any peripheral interrupt, external interrupt or WKPU source.

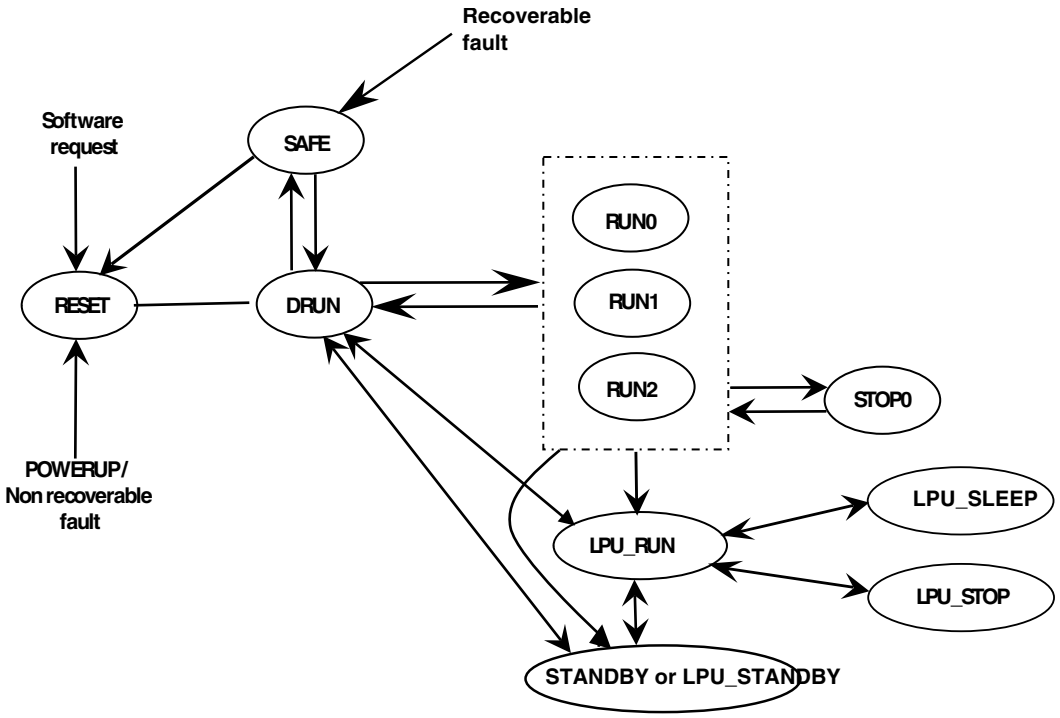


Figure 2-2. Mode relationship

NOTE

If we are entering into complete power gated mode via LPU_RUN mode, it is called LPU_STANDBY else it is referred as STANDBY. Only difference in two modes is that user can select clock sources to be ON/OFF by configuring ME_STANDBY0_MC for STANDBY mode and LPU_STANDBY_CF for LPU_STANDBY mode

When exiting STANDBY or LPU_RUN, the user will determine which core(s) are active. This will be optionally the Z2 or Z4.

Brief Description of Power Domains

The Power Control Unit (MC_PCU) and Low Power Subsystem LPU are used to manage all mode transition. Power can be saved by gating the power supply via a power switching device. The device is grouped into 3 parts which are called “power domains”. When a power domain is disconnected from the supply, the power consumption is reduced to zero in that domain. Any status information of such a power domain is lost.

Power Domain0 (PD0): It is an always alive domain, which consists of modules required for mode transitions and some timers.

Power Domain1 (PD1): It consists of modules required to be operational in LPU modes. z2 core is present in PD1. This domain is power gated off in STANDBY/ LPU_STANDBY mode.

Power Domain2 (PD2): It is the biggest domain with most of the modules, many XBAR masters. z4 core is present in PD2. This domain is power gated off in any of the LPU modes as well as STANDBY mode.

Powered with External Regulator

If the application is using an external 1.25 V supply, then that 1.25 V supply will continue to be connected to the PD2 domain unless the customer application takes steps to remove it. Hence depending on the application configuration there may be leakage from the PD2 area in STANDBY or LPU modes. This can be avoided by disconnecting the external supply when the device has entered the low power states, although it must be re-applied to enable exit from the low power states. For details, see Applications and Low Power Modes.

Maximum power saving is reached by entering the STANDBY modes. On each mode change request, the power and mode controllers in the chip i.e. MC_ME, MC_PCU, LPU_CTL and PMC evaluates the power domain settings in the power domain configuration registers and initiates a power-down or a power-up sequence for each individual power domain. The power-up/down sequences are handled by finite state machines to ensure a smooth and safe transition from one power state to the other.

Low Power Mode Exit

Exiting STANDBY modes can only be done via a system wakeup event or reset as all power domains other than PD0 are in the power-down state. Refer the wakeup sources for the wakeup event triggers details. See the device I/O Signal Description and Input Multiplexing tables attached to this document as an Excel file. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click the Excel file to open it and select the I/O Signal Description table tab.

NOTE

For interrupt handling during mode transitions, refer to section: Interrupts handling during mode transitions.

Chapter 3
Memory Map

3.1 Introduction

This chapter describes the memory and peripheral locations within the specified memory space. The addresses represent the physical addresses assigned to each region or module. An access to the reserved locations may produce undesirable results.

3.2 System memory map

The following table shows the system memory map.

Table 3-1. System memory map

Start Address	End Address	Size	Description
0x00000000	0x003FFFFF	16 KB	Reserved
0x00400000	0x00403FFF	16 KB	UTEST
0x00404000	0x00407FFF	16 KB	BAF
0x00408000	0x0060FFFF	2080 KB	Reserved
0x00610000	0x0062FFFF	128 KB	HSM code
0x00630000	0x00F7FFFF	9536 KB	Reserved
0x00F80000	0x00F87FFF	32 KB	HSM data
0x00F88000	0x00F8BFFF	16 KB	Reserved
0x00F8C000	0x00F8FFFF	16 KB	Small HSM Code Block
0x00F90000	0x00FAFFFF	128 KB	Small and Medium NVM Blocks
0x00FB0000	0x00FBFFFF	64 KB	Reserved
0x00FC0000	0x00FFFFFF	256 KB	Small & Medium NVM Blocks
0x01000000	0x0127FFFF	2560 KB	Large Flash NVM
0x01280000	0x0157FFFF	3072 KB	Reserved
0x01780000	0x3FFFFFFF	1000 MB	Reserved
0x40000000	0x4007FFFF	512 KB	System RAM

Table continues on the next page...

Table 3-1. System memory map (continued)

Start Address	End Address	Size	Description
0x400C0000	0xEFFFFFFF	2815 MB	Reserved
0xF0000000	0xFFFFFFFF	256 MB	Peripheral Address Space

3.3 NVM memory map

Table 3-2. NVM memory map

Start Address	End Address	Allocated Size (KB/MB)	Complete Flash Block Structure	RWW Partition	Block Size (KB)
0x00000000	0x003FFFFF	4096 KB			Reserved
UTEST NVM Block					
0x00400000	0x00403FFF	16 KB	UTEST	0	16
BAF Block					
0x00404000	0x00407FFF	16 KB	BAF	1	16
0x00408000	0x0060FFFF	2080 KB			Reserved
HSM Code					
0x00610000	0x0061FFFF	64 KB	HSM code block 2	0	64
0x00620000	0x0062FFFF	64 KB	HSM code block 3	1	64
0x00630000	0x00F7FFFF	9536 KB			Reserved
HSM Data					
0x00F80000	0x00F83FFF	16 KB	HSM data block0	4	16
0x00F84000	0x00F87FFF	16 KB	HSM data block1	5	16
0x00F88000	0x00F8BFFF	16 KB			Reserved
Small HSM Code Block					
0x00F8C000	0x00F8FFFF	16 KB	16KB Code Flash block	0	16
Small and Medium Flash Blocks					
0x00F90000	0x00F93FFF	16 KB	16KB Code Flash block	2	16
0x00F94000	0x00F97FFF	16 KB	16KB Code Flash block	2	16
0x00F98000	0x00F9BFFF	16 KB	16KB Code Flash block	2	16
0x00F9C000	0x00F9FFFF	16 KB	16KB Code Flash block	2	16
0x00FA0000	0x00FA3FFF	16KB	16KB Code Flash block	3	16
0x00FA4000	0x00FA7FFF	16 KB	16KB Code Flash block	3	16
0x00FA8000	0x00FABFFF	16 KB	16KB Code Flash block	3	16

Table continues on the next page...

Table 3-2. NVM memory map (continued)

Start Address	End Address	Allocated Size (KB/MB)	Complete Flash Block Structure	RWW Partition	Block Size (KB)
0x00FAC000	0x00FAFFFF	16 KB	16KB Code Flash block	3	16
0x00FB0000	0x00FB7FFF	32 KB	Reserved		
0x00FB8000	0x00FBFFFF	32 KB	Reserved		
0x00FC0000	0x00FC7FFF	32 KB	32KB Code Flash block	0	32
0x00FC8000	0x00FCFFFF	32 KB	32KB Code Flash block	0	32
0x00FD0000	0x00FD7FFF	32 KB	32KB Code Flash block	1	32
0x00FD8000	0x00FDFFFF	32 KB	32KB Code Flash block	1	32
0x00FE0000	0x00FEFFFF	64 KB	64KB Code Flash block	0	64
0x00FF0000	0x00FFFFFF	64 KB	64KB Code Flash block	1	64
Large Flash Blocks					
0x01000000	0x0103FFFF	256 KB	256KB Code Flash block0	6	256
0x01040000	0x0107FFFF	256 KB	256KB Code Flash block1	6	256
0x01080000	0x010BFFFF	256 KB	256KB Code Flash block2	6	256
0x010C0000	0x010FFFFF	256 KB	256KB Code Flash block3	6	256
0x01100000	0x0113FFFF	256 KB	256KB Code Flash block4	6	256
0x01140000	0x0117FFFF	256 KB	256KB Code Flash block5	7	256
0x01180000	0x011BFFFF	256 KB	256KB Code Flash block6	7	256
0x011C0000	0x011FFFFF	256 KB	256KB Code Flash block7	7	256
0x01200000	0x0123FFFF	256 KB	256KB Code Flash block8	7	256
0x01240000	0x0127FFFF	256 KB	256KB Code Flash block9	7	256
0x01280000	0x012BFFFF	256 KB	Reserved		
0x012C0000	0x012FFFFF	256 KB	Reserved		
0x01300000	0x0133FFFF	256 KB	Reserved		
0x01340000	0x0137FFFF	256 KB	Reserved		
0x01380000	0x013BFFFF	256 KB	Reserved		
0x013C0000	0x013FFFFF	256 KB	Reserved		
0x01400000	0x0143FFFF	256 KB	Reserved		

Table continues on the next page...

Table 3-2. NVM memory map (continued)

Start Address	End Address	Allocated Size (KB/MB)	Complete Flash Block Structure	RWW Partition	Block Size (KB)
0x01440000	0x0147FFFF	256 KB	Reserved		
0x01480000	0x014BFFFF	256 KB	Reserved		
0x014C0000	0x014FFFFF	256 KB	Reserved		
0x01500000	0x0153FFFF	256 KB	Reserved		
0x01540000	0x0157FFFF	256 KB	Reserved		
0x01580000	0x0179FFFF	2176 KB	Reserved		

3.4 UTEST Memory Map

Table 3-3. UTEST Memory Map

Start Address	End Address	Allocated Size [bytes]	Decription
UTEST Flash			
0x00400000	0x00400001	2	Reserved
0x00400002	0x00400003	2	Reserved
0x00400004	0x00400005	2	Reserved
0x00400006	0x00400007	2	Reserved
0x00400008	0x0040000B	4	Reserved
0x0040000C	0x0040000F	4	Test Mode Disable Seal
0x00400010	0x0040001F	16	Test Mode Disable Block Select Group A
0x00400020	0x0040002F	16	Factory Erase diary Location
0x00400030	0x0040003F	16	Test Mode Disable Block Select Group B
0x00400040	0x0040005F	32	Customer Single Bit Correction Area
0x00400060	0x0040007F	32	Customer Double Bit Detection Area
0x00400080	0x0040009F	32	Customer EDC after ECC Area
0x004000A0	0x004000BF	32	UID
0x004000C0	0x004000C3	4	Soft DCF Record Start Address
0x004000C4	0x004000C7	4	Reserved
0x004000C8	0x004000FF	56	Reserved
0x00400100	0x00400103	4	Test Mode Overide Passcode
0x00400104	0x0040011F	28	Reserved
0x00400120	0x0040013F	32	JTAG Password
0x00400140	0x0040015F	32	PASS Password Group 0

Table continues on the next page...

Table 3-3. UTEST Memory Map (continued)

Start Address	End Address	Allocated Size [bytes]	Decription
0x00400160	0x0040017F	32	PASS Password Group 1
0x00400180	0x0040019F	32	PASS Password Group 2
0x004001A0	0x004001BF	32	PASS Password Group 3
0x004001C0	0x004001DF	32	Reserved
0x004001E0	0x004001FF	32	Reserved
0x00400200	0x0040020F	16	Lifecycle Slot 0 - MCU_PROD
0x00400210	0x0040021F	16	Lifecycle Slot 1 - CUST_DEL
0x00400220	0x0040022F	16	Lifecycle Slot 2 - OEM_PROD
0x00400230	0x0040023F	16	Lifecycle Slot 3 - IN_FIELD
0x00400240	0x0040024F	16	Lifecycle Slot 4 - FA
0x00400250	0x004002FF	176	Reserved
0x00400300	0x00400307	8	DCF Start Record
0x00400308 [1, 2]	0x00400347	64	Reserved
0x00400348	0x00400FFF	3256	DCF Records
0x00401000	0x00403FFF	12288	Reserved for customer OTP data

1. When device is in CUST_DEL (or later) life cycle and prefetching is enabled in the platform flash controller (PFLASH_C55FM), it reads of the HSM 'ROM' key locations (0x0400308 - 0x0400347) and return dummy DCF records. However, accesses to these locations may cause result in anomalous operation. Reading the HSM 'ROM' key locations can result in a false ECC event, specifically either a false single-bit correction or a false multi-bit error detection. Reading the HSM 'ROM' key locations can also interfere with an unrelated, but coincident, loading of the PFLASH_C55FM prefetch buffers. As a result a subsequent request to flash, may result in a false ECC event, specifically either a false single-bit correction or a false multi-bit error detection.
2. Program MPU to prevent accesses to 0x0400308 - 0x0400347. Avoid accessing the HSM ROM key locations in the flash UTEST region (0x0400308 - 0x0400347). Prior to accessing this region; disable flash buffer prefetches.

3.5 RAM memory map

Table 3-4. RAM memory map

Start Address	End Address	Allocated Size [KB]	Description
0x40000000	0x40001FFF	8 (STANDBY)	SRAM 0
0x40002000	0x4000FFFF	56	SRAM 1
0x40010000	0x4001FFFF	64	SRAM 2
0x40020000	0x4002FFFF	64	SRAM 3
0x40030000	0x4003FFFF	64	SRAM 4
0x40040000	0x4004FFFF	64	SRAM 5
0x40050000	0x4005FFFF	64	SRAM 6
0x40060000	0x4006FFFF	64	SRAM 7
0x40070000	0x4007FFFF	64	SRAM 8
0x40080000	0x400BFFFF	Reserved	-

3.6 Peripheral Bridges A and B memory maps

3.6.1 Peripheral Bridge A memory map

Table 3-5. Peripheral Bridge A memory map

Start Address	End Address	Size [KB]	ME_ PCTL	Region	Peripheral Bridge slot number
On Platform Bridge A					
0xFC000000	0xFC003FFF	16 KB		PBRIDGE_A	PACR0
0xFC004000	0xFC007FFF	16 KB		XBAR_0	PACR1
0xFC008000	0xFC00BFFF	16 KB		Reserved	
0xFC00C000	0xFC00FFFF	16 KB		Reserved	
0xFC010000	0xFC013FFF	16 KB		SMPU_0	PACR4
0xFC014000	0xFC017FFF	16 KB		Reserved	
0xFC018000	0xFC01FFFF	16 KB		Reserved	
0xFC01C000	0xFC01FFFF	16 KB		Reserved	
0xFC020000	0xFC023FFF	16 KB		PRAM_0	PACR8
0xFC024000	0xFC027FFF	16 KB		PRAM_1	PACR9
0xFC028000	0xFC02BFFF	16 KB		PCM	PACR10
0xFC02C000	0xFC02FFFF	16 KB		Reserved	
0xFC030000	0xFC033FFF	16 KB		PFLASH	PACR12
0xFC034000	0xFC037FFF	16 KB		Reserved	
0xFC038000	0xFC03BFFF	16 KB			
0xFC03C000	0xFC03FFFF	16 KB		SEMA4	PACR15
0xFC040000	0xFC043FFF	16 KB		INTC	PACR16
0xFC044000	0xFC047FFF	16 KB		Reserved	
0xFC048000	0xFC04BFFF	16 KB		Reserved	
0xFC04C000	0xFC04FFFF	16 KB		Reserved	
0xFC050000	0xFC053FFF	16 KB		SWT_0	PACR20
0xFC054000	0xFC057FFF	16 KB		SWT_1	PACR21
0xFC058000	0xFC05BFFF	16 KB		Reserved	
0xFC05C000	0xFC05CFFFF	16 KB		Reserved	
0xFC060000	0xFC063FFF	16 KB		Reserved	
0xFC064000	0xFC067FFF	16 KB		Reserved	
0xFC068000	0xFC06BFFF	16 KB		STM_0	PACR26
0xFC06C000	0xFC06FFFF	16 KB		STM_1	PACR27
0xFC070000	0xFC09BFFF	176 KB		Reserved	
0xFC0A0000	0xFC0A3FFF	16 KB		eDMA	PACR40
0xFC0A4000	0xFCFFFFFF	15 MB		Reserved	

Table continues on the next page...

Table 3-5. Peripheral Bridge A memory map (continued)

Start Address	End Address	Size [KB]	ME_ PCTL	Region	Peripheral Bridge slot number
Off Platform Bridge A					
0xFD000000	0xFFC0FFFF			Reserved	
0xFFC10000	0xFFC13FFF	16 KB	0	BCTU	OPACR251
0xFFC14000	0xFFC17FFF	16 KB		Reserved	
0xFFC18000	0xFFC1BFFF	16 KB		Reserved	
0xFFC1C000	0xFFC1FFFF	16 KB		Reserved	
0xFFC20000	0xFFC23FFF	16 KB		GLITCH_FILTER_1	OPACR247
0xFFC24000	0xFFC27FFF	16 KB		GLITCH_FILTER_0	OPACR246
0xFFC28000	0xFFC2BFFF	16 KB		Reserved	
0xFFC2C000	0xFFC2FFFF	16 KB		Reserved	
0xFFC30000	0xFFC33FFF	16 KB		HFIF_1	OPACR243
0xFFC34000	0xFFC37FFF	16 KB		HFIF_0	OPACR242
0xFFC38000	0xFFC3BFFF	16 KB	1	eMIOS_0	OPACR241
0xFFC3C000	0xFFC3FFFF	16 KB		Reserved	
0xFFC40000	0xFFC8FFFF	320 KB		Reserved	
0xFFC90000	0xFFC93FFF	16 KB		Reserved	
0xFFC94000	0xFFC97FFF	16 KB		Reserved	
0xFFC98000	0xFFC9BFFF	16 KB		Reserved	
0xFFC9C000	0xFFC9FFFF	16 KB		Reserved	
0xFFCA0000	0xFFCA3FFF	16 KB		Reserved	
0xFFCA4000	0xFFCA7FFF	16 KB		Reserved	
0xFFCA8000	0xFFCB3FFF	48 KB		Reserved	
0xFFCB4000	0xFFCB7FFF	16 KB	10	SAI_0	OPACR210
0xFFCB8000	0xFFCBBFFF	16 KB	12	SAI_2	OPACR209
0xFFCBC000	0xFFCC3FFF	32 KB		Reserved	
0xFFCC4000	0xFFCC7FFF	16 KB	15	ENET 0	OPACR206
0xFFCC8000	0xFFCCBFFF	16 KB		Reserved	
0xFFCCC000	0xFFCCFFFF	16 KB		Reserved	
0xFFCD0000	0xFFCD3FFF	16 KB		Reserved	
0xFFCD4000	0xFFCDBFFF	32 KB		Reserved	
0xFFCDC000	0xFFCDFFFF	16 KB	20	ACMP_0	OPACR200
0xFFCE0000	0xFFCE3FFF	16 KB	21	ACMP_1	OPACR199
0xFFCE4000	0xFFCE7FFF	16 KB	22	ACMP_2	OPACR198
0xFFCE8000	0xFFDFFFFF	1 MB		Reserved	
0xFFE00000	0xFFE03FFF	16 KB	24	ADC_0	OPACR127
0xFFE04000	0xFFE4FFFF	304 KB		Reserved	
0xFFE50000	0xFFE53FFF	16 KB	28	FlexRay	OPACR107
0xFFE54000	0xFFE67FFF	80 KB		Reserved	
0xFFE68000	0xFFE6BFFF	16 KB	30	I^2C_0	OPACR101

Table continues on the next page...

Table 3-5. Peripheral Bridge A memory map (continued)

Start Address	End Address	Size [KB]	ME_ PCTL	Region	Peripheral Bridge slot number
0xFFE6C000	0xFFE6FFFF	16 KB	32	I^2C_2	OPACR100
0xFFE70000	0xFFE73FFF	16 KB	40	DSPI_0	OPACR99
0xFFE74000	0xFFE77FFF	16 KB	42	DSPI_2	OPACR98
0xFFE78000	0xFFE8BFFF	80 KB		Reserved	
0xFFE8C000	0xFFE8FFFF	16 KB	50	LIN_0	OPACR92
0xFFE90000	0xFFE93FFF	16 KB	52	LIN_2	OPACR91
0xFFE94000	0xFFE97FFF	16 KB	54	LIN_4	OPACR90
0xFFE98000	0xFFE9BFFF	16 KB	56	LIN_6	OPACR89
0xFFE9C000	0xFFE9FFFF	16 KB	58	LIN_8	OPACR88
0xFFEA0000	0xFFEA3FFF	16 KB	60	LIN_10	OPACR87
0xFFEA4000	0xFFEA7FFF	16 KB	62	LIN_12	OPACR86
0xFFEA8000	0xFFEABFFF	16 KB	64	LIN_14	OPACR85
0xFFEAC000	0xFFEBFFFF	80 KB		Reserved	
0xFFEC0000	0xFFEC3FFF	16 KB	70	FLEXCAN_0	OPACR79
0xFFEC4000	0xFFEC7FFF	16 KB	72	FLEXCAN_2	OPACR78
0xFFEC8000	0xFFECBFFF	16 KB	74	FLEXCAN_4	OPACR77
0xFFECC000	0xFFECFFFF	16 KB	76	FLEXCAN_6	OPACR76
0xFFED0000	0xFFF2BFFF	368 KB		Reserved	
0xFFF2C000	0xFFF2FFFF	16 KB		HSM_SWT	OPACR52
0xFFF30000	0xFFF33FFF	16 KB		HSM	OPACR51
0xFFF34000	0xFFF37FFF	16 KB		TDM	OPACR50
0xFFF38000	0xFFF3BFFF	16 KB		Reserved	
0xFFF3C000	0xFFF3FFFF	16 KB	82	JDC (JTAG Data Communication)	OPACR48
0xFFF40000	0xFFF43FFF	16 KB		Reserved	
0xFFF44000	0xFFF47FFF	16 KB		STCU	OPACR46
0xFFF48000	0xFFF4BFFF	16 KB		Reserved	
0xFFF4C000	0xFFF4FFFF	16 KB		Reserved	
0xFFF50000	0xFFF53FFF	16 KB	83	MEMU_0	OPACR43
0xFFF54000	0xFFF57FFF	16 KB		Reserved	
0xFFF58000	0xFFF5BFFF	16 KB		Reserved	
0xFFF5C000	0xFFF5FFFF	16 KB		Reserved	
0xFFF60000	0xFFF63FFF	16 KB		Reserved	
0xFFF64000	0xFFF67FFF	16 KB	84	CRC	OPACR38
0xFFF68000	0xFFF6BFFF	16 KB		Reserved	
0xFFF6C000	0xFFF6FFFF	16 KB	90	DMAMUX	OPACR36
0xFFF70000	0xFFF73FFF	16 KB		Reserved	
0xFFF74000	0xFFF77FFF	16 KB		Reserved	
0xFFF78000	0xFFF7BFFF	16 KB		Reserved	

Table continues on the next page...

Table 3-5. Peripheral Bridge A memory map (continued)

Start Address	End Address	Size [KB]	ME_ PCTL	Region	Peripheral Bridge slot number
0xFFF7C000	0xFFF7FFFF	16 KB		Reserved	
0xFFF80000	0xFFF83FFF	16 KB		Reserved	
0xFFF84000	0xFFF87FFF	16 KB	91	PIT_RTI	OPACR30
0xFFF88000	0xFFF8BFFF	16 KB		Reserved	
0xFFF8C000	0xFFF8FFFF	16 KB		Reserved	
0xFFF90000	0xFFF93FFF	16 KB		Reserved	
0xFFF94000	0xFFF97FFF	16 KB		GPR (General purpose registers)	OPACR26
0xFFF98000	0xFFF9BFFF	16 KB	93	WKPU	OPACR25
0xFFF9C000	0xFFF9FFFF	16 KB		LPU_CTL	OPACR24
0xFFFA0000	0xFFFA3FFF	16 KB		MC_PCU	OPACR23
0xFFFA4000	0xFFFA7FFF	16 KB		Reserved	
0xFFFA8000	0xFFFABFFF	16 KB		MC_RGM	OPACR21
0xFFFAC000	0xFFFAFFFF	16 KB		Reserved	
0xFFFB0000	0xFFFB3FFF	16 KB		MC_CGM	OPACR19
0xFFFB4000	0xFFFB7FFF	16 KB		Reserved	
0xFFFB8000	0xFFFBBFFF	16 KB		MC_ME	OPACR17
0xFFFBC000	0xFFFBFFFF	16 KB		Reserved	
0xFFFC0000	0xFFFC3FFF	16 KB		SIUL	OPACR15
0xFFFC4000	0xFFFC7FFF	16 KB		Reserved	
0xFFFC8000	0xFFFCBFFF	16 KB		Reserved	
0xFFFCC000	0xFFFCFFFF	16 KB	96	SPI_0	OPACR12
0xFFFD0000	0xFFFD3FFF	16 KB	98	SPI_2	OPACR11
0xFFFD4000	0xFFFD7FFF	16 KB		Reserved	
0xFFFD8000	0xFFFDBFFF	16 KB		Reserved	
0xFFFDC000	0xFFFDFFFF	16 KB		Reserved	
0xFFFE0000	0xFFFE3FFF	16 KB		C55FMC (Flash main control)	OPACR7
0xFFFE4000	0xFFFE7FFF	16 KB		ALT_C55FMC[1] (Flash alternate control)	OPACR6
0xFFFE8000	0xFFFEBFFF	16 KB		Reserved	
0xFFFEC000	0xFFFEFFFF	16 KB	105	MEMU_1	OPACR4
0xFFFF0000	0xFFFF3FFF	16 KB	102	RTC-API	OPACR3
0xFFFF4000	0xFFFF7FFF	16 KB		PASS	OPACR2
0xFFFF8000	0xFFFFBFFF	16 KB		SSCM	OPACR1
0xFFFFC000	0xFFFFFFFF	16 KB		Reserved	

1. For ALT_C55FMC please refer the "MPC5746C Microcontroller Security Reference Manual"

3.6.2 Peripheral Bridge B memory map

Table 3-6. Peripheral Bridge B memory map

Start Address	End Address	Size [KB]	ME_ PCTL	Region	Peripheral Bridge slot number
On Platform Bridge B					
0xF8000000	0xF8003FFF	16	.	PBRIDGE_B	PACR0
0xF8004000	0xF8FFFFFF	16 MB		Reserved	
Off Platform Bridge B					
0xF9000000	0xFBC17FFF	44 MB		Reserved	
0xFBC18000	0xFBC1BFFF	16 KB		Glitch_Filter_3	OPACR249
0xFBC1C000	0xFBC1FFFF	16 KB		Glitch_Filter_2	OPACR248
0xFBC20000	0xFBC23FFF	16 KB		Reserved	
0xFBC24000	0xFBC27FFF	16 KB		Reserved	
0xFBC28000	0xFBC2BFFF	16 KB		HFIF_3	OPACR245
0xFBC2C000	0xFBC2FFFF	16 KB		HFIF_2	OPACR244
0xFBC30000	0xFBC33FFF	16 KB		Reserved	
0xFBC34000	0xFBC37FFF	16 KB		Reserved	
0xFBC38000	0xFBCB3FFF	16 KB	2	eMIOS_1	OPACR241
0xFBC3C000	0xFBCB3FFF	16 KB		Reserved	
0xFBCB4000	0xFBCB7FFF	16 KB	11	SAI_1	OPACR210
0xFBCB8000	0xFBDFFFFF	1312 KB		Reserved	
0xFBE00000	0xFBE03FFF	16 KB	25	ADC_1	OPACR127
0xFBE04000	0xFBE67FFF	400 KB		Reserved	
0xFBE68000	0xFBE6BFFF	16 KB	31	I^2C_1	OPACR101
0xFBE6C000	0xFBE6FFFF	16 KB	33	I^2C_3	OPACR100
0xFBE70000	0xFBE73FFF	16 KB	41	DSPI_1	OPACR99
0xFBE74000	0xFBE77FFF	16 KB	43	DSPI_3	OPACR98
0xFBE78000	0xFBE8BFFF	80 KB		Reserved	
0xFBE8C000	0xFBE8FFFF	16 KB	51	LIN_1	OPACR92
0xFBE90000	0xFBE93FFF	16 KB	53	LIN_3	OPACR91
0xFBE94000	0xFBE97FFF	16 KB	55	LIN_5	OPACR90
0xFBE98000	0xFBE9BFFF	16 KB	57	LIN_7	OPACR89
0xFBE9C000	0xFBE9FFFF	16 KB	59	LIN_9	OPACR88
0xFBEA0000	0xFBEA3FFF	16 KB	61	LIN_11	OPACR87
0xFBEA4000	0xFBEA7FFF	16 KB	63	LIN_13	OPACR86
0xFBEA8000	0xFBEABFFF	16 KB	65	LIN_15	OPACR85
0xFBEAC000	0xFBEAFFFF	16 KB		Reserved	
0xFBEB0000	0xFBEBFFFF	64 KB		Reserved	
0xFBEC0000	0xFBEC3FFF	16 KB	71	FLEXCAN_1	OPACR79

Table continues on the next page...

Table 3-6. Peripheral Bridge B memory map (continued)

Start Address	End Address	Size [KB]	ME_ PCTL	Region	Peripheral Bridge slot number
0xFBEC4000	0xFBEC7FFF	16 KB	73	FLEXCAN_3	OPACR78
0xFBEC8000	0xFBECBFFF	16 KB	75	FLEXCAN_5	OPACR77
0xFBECC000	0xFBECFFFF	16 KB	77	FLEXCAN_7	OPACR76
0xFBED0000	0xFBF57FFF	544 KB		Reserved	
0xFBF58000	0xFBF5BFFF	16 KB		FCCU	OPACR41
0xFBF5C000	0xFBF9BFFF	256 KB		Reserved	
0xFBF9C000	0xFBF9FFFF	16 KB	97	SPI_1	OPACR24
0xFBFA0000	0xFBFA3FFF	16 KB	99	SPI_3	OPACR23
0xFBFA4000	0xFBFA7FFF	16 KB		Reserved	
0xFBFA8000	0xFBFAFFFF	32 KB		Reserved	
0xFBFB0000	0xFBFB3FFF	16 KB		Reserved	
0xFBFB4000	0xFBFFFFFF	304 KB		Reserved	

3.7 MC_CGM mapped peripherals memory map

Table 3-7. MC_CGM mapped peripherals memory map

Start Address	End Address	Size [Bytes]	Mapped Peripheral
0xFFFB_0000	0xFFFB_003F	64	FIRC
0xFFFB_0040	0xFFFB_007F	64	FXOSC
0xFFFB_0080	0xFFFB_00BF	64	PLLDIG
0xFFFB_00C0	0xFFFB_00FF	64	SIRC
0xFFFB_0100	0xFFFB_013F	64	SXOSC
0xFFFB_0140	0xFFFB_017F	64	CMU
0xFFFB_0180	0xFFFB_6BC0	1084	CGM

3.8 PCU mapped peripherals memory map

Table 3-8. PCU mapped peripherals memory map

Start Address	End Address	Size [Bytes]	Mapped Peripheral
0xFFFA_0400	0xFFFA_07FF	1024	PMCDIG

3.9 DMAMUX mapped peripherals memory map

Table 3-9. DMAMUX mapped peripherals memory map

Start Address	End Address	Size [Bytes]	Mapped Peripheral
0xFFF6_C000	0xFFF6_C1FF	512	DMAMUX_0
0xFFF6_C200	0xFFF6_C3FF	512	DMAMUX_1
0xFFF6_C400	0xFFF6_FFF	15 KB	Reserved

Chapter 4
Signal Description

4.1 Generic pins/balls

The I/O Signal Description Table contains information on generic pins/balls. See the device I/O Signal Description and Input Multiplexing tables attached to this document as an Excel file. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click the Excel file to open it and select the I/O Signal Description table tab.

NOTE

On 176LQFP-EP package, expose pad must be connected to VSS_LV on board.

4.1.1 MSCR assignments

The I/O Signal Description table provides the port pin, SIUL Multiplexed Signal Configuration Register (MSCR) number, MSCR Source Signal Select (SSS) function, module, signal description, direction, pad type, and package pin/ball numbers for each I/O pin on the device.

The Implemented SIUL2 Multiplexed Single Configuration Register details is provided in the I/O Signal Description Table attached as excel.

NOTE

Implemented SIUL2 Multiplexed Single Configuration Register are from SIUL2_MSCR0 to SIUL2_MSCR263 and SIUL2_IMCR0 to SIUL2_IMCR511. Please note that MSCR 513 in this column corresponds to IMCR 0 and so on.

Example:

If user wants to configure CAN0RX and CAN0TX, then the following configuration can be used in SIUL2 registers:

```
SIUL2.MSCR[16].R  = 0x03000001; /* PB[0] -&gt OBE = 1, ODE = 1 and MSCR16_SSS = 1 */
SIUL2.MSCR[17].R  = 0x00080000; /* PB[1] -&gt IBE = 1 */
SIUL2.IMCR[188].R = 0x00080002; /* PB[1] -&gt MSCR17_SSS = 2, notice that
"IO_Signal_Description_and_Input_multiplexing_tables.xls" shows 700 for the MSCR/IMCR see
note above, thus 188 must be used to properly configure the pin signal connected to it */
```

MSCR bit fields corresponds to pin/pad basis, these are independent of muxing implemented on that specific pin/pad. As an example, PAD18/PB[2] has SRE, ODE, SMC, IBE, HYS, PUS, PUE, SSS and OBE are implemented with the reset value 0000_0000.

In the “IO_Signal_Description_and_Input_multiplexing_tables.xls" spreadsheet, in "IO Signal Table" worksheet, the MSCR register bits shows ‘0’, ‘1’, and ‘-’ for state of all bits associated with different ports. ‘0’ or ’1’ represents the reset state and ‘-’ represents the bit is not supported on the respective port.

NOTE

IF HSM DO functionality is used on GPIO[28], GPIO[60] pads or GPIO[134], GPIO[135] pads using HSM_IO configurations for security, then all four GPIOs would get dedicated to HSM.

4.2 PAD keeping feature in LPU/STANDBY mode

The device is capable to retain the output value of some selected pads when device is power gated i.e. in LPU or STANDBY mode. These pads reside in always alive power domain(PD0).

To enable this feature, set PMCDIG_RDCR[PAD_KEEP_EN] to '1'. The PAD's output values are retained until software writes a '0' to this bit on exit from LPU or STANDBY mode . If the bit is not set, latched PAD values are automatically cleared on LPU or STANDBY exit and the PADS go to Hi-z

Table 4-1 lists the pads which support PAD Keeping feature. However, the pads in the following two tables combine to form the full list of pads available in LPU and STANDBY modes.

Table 4-1. PADS alive in LPU mode with pad keeping feature

Function	I/O Power Segment
GPIO[12]	STBY_LPU
GPIO[13]	STBY_LPU
GPIO[14]	STBY_LPU
GPIO[16]	STBY_LPU
GPIO[27]	STBY_LPU
GPIO[28]	STBY_LPU

Table continues on the next page...

Table 4-1. PADS alive in LPU mode with pad keeping feature (continued)

Function	I/O Power Segment
GPIO[32]	STBY_LPU
GPIO[33]	STBY_LPU
GPIO[38]	STBY_LPU
GPIO[42]	STBY_LPU
GPIO[60]	STBY_LPU
GPIO[61]	STBY_LPU
GPIO[80]	STBY_LPU
GPIO[81]	STBY_LPU
GPIO[82]	STBY_LPU
GPIO[83]	STBY_LPU
GPIO[84]	STBY_LPU
GPIO[85]	STBY_LPU
GPIO[86]	STBY_LPU
GPIO[87]	STBY_LPU
GPIO[88]	STBY_LPU
GPIO[90]	STBY_LPU
GPIO[92]	STBY_LPU
GPIO[102]	STBY_LPU
GPIO[104]	STBY_LPU
GPIO[121]	STBY_LPU
GPIO[122]	STBY_LPU
GPIO[137]	STBY_LPU
GPIO[138]	STBY_LPU
GPIO[142]	STBY_LPU
GPIO[143]	STBY_LPU
GPIO[144]	STBY_LPU
GPIO[145]	STBY_LPU
GPIO[146]	STBY_LPU
GPIO[147]	STBY_LPU
GPIO[149]	STBY_LPU
GPIO[150]	STBY_LPU
GPIO[151]	STBY_LPU
GPIO[152]	STBY_LPU
GPIO[153]	STBY_LPU
GPIO[154]	STBY_LPU
GPIO[155]	STBY_LPU
GPIO[156]	STBY_LPU
GPIO[158]	STBY_LPU
GPIO[159]	STBY_LPU

Table continues on the next page...

Table 4-1. PADS alive in LPU mode with pad keeping feature (continued)

Function	I/O Power Segment
GPIO[160]	STBY_LPU
GPIO[161]	STBY_LPU
GPIO[162]	STBY_LPU
GPIO[163]	STBY_LPU
GPIO[164]	STBY_LPU
GPIO[165]	STBY_LPU
GPIO[166]	STBY_LPU
GPIO[167]	STBY_LPU
GPIO[168]	STBY_LPU
GPIO[197]	STBY_LPU
GPIO[198]	STBY_LPU
GPIO[205]	STBY_LPU
GPIO[206]	STBY_LPU

NOTE

For pads 42 and 102, only if source signal select is configured for CMP0_O and CLKOUT1 respectively via SIUL2_MSCR registers, the analog comparator 0 CMP0_O and CLKOUT1 clock would be exposed during standby. If expose during standby is enabled for CMP0_O or CLKOUT1 on pad 42 and 102 respectively, then padkeeping on these pads would be disabled.

Table 4-2. PADS used for WKPU

Function	I/O power segment
GPIO[0]	LPU_WK
GPIO[1]	LPU_WK
GPIO[2]	LPU_WK
GPIO[4]	LPU_WK
GPIO[15]	LPU_WK
GPIO[17]	LPU_WK
GPIO[19]	LPU_WK
GPI[24]	LPU_WK
GPI[25]	LPU_WK
GPIO[26]	LPU_WK
GPIO[39]	LPU_WK
GPIO[41]	LPU_WK
GPIO[43]	LPU_WK

Table continues on the next page...

Table 4-2. PADS used for WKPU (continued)

Function	I/O power segment
GPI[48]	LPU_WK
GPI[49]	LPU_WK
GPIO[64]	LPU_WK
GPIO[67]	LPU_WK
GPIO[69]	LPU_WK
GPIO[73]	LPU_WK
GPIO[75]	LPU_WK
GPIO[89]	LPU_WK
GPIO[91]	LPU_WK
GPIO[93]	LPU_WK
GPIO[99]	LPU_WK
GPIO[101]	LPU_WK
GPIO[103]	LPU_WK
GPIO[105]	LPU_WK
GPIO[129]	LPU_WK
GPIO[131]	LPU_WK
GPIO[157]	LPU_WK

Chapter 5
Security Overview

5.1 Introduction

All MPC574xx family devices have a comprehensive set of customer-configurable security features designed to protect code and data from unauthorized access. Some security features are available in all device configurations. Other, more advanced security features are not available in all device configurations. These more advanced features require more than simply being turned on or off. They require significant involvement for implementation; consequently, they provide an important opportunity to define the security level of the device.

5.2 Basic security

All MPC574xx devices have the following basic set of security features.

- Device censorship based on life cycle model. Access to device code and data becomes progressively more restricted as the device matures through a defined set of life cycle steps.
- Memory security features
  - NVM censorship support, password protection, and one-time-programmable (OTP) flash memory areas, Flash erase counter and tamper detection.
- Unique ID for each device. Each MPC574xx device has a unique identification number stored in an OTP flash memory area. This unique ID can be read by any of the cores on the device.
- Basic debugger restrictions (on/off via censorship mode)

5.3 Advanced security

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

- Customer programmable Hardware Security Module (HSM). The HSM is a dedicated security subsystem that includes a processor core, dedicated SRAM, an encryption module, and exclusive access to secure areas of device flash memory. The HSM runs code independently from the main device processor cores and can be used to implement advanced security and monitoring functions.
- Secure debugger interface
- Boot modes
  - Trusted/secure boot support.
  - Handshake with Boot Assist Flash (BAF) supported.
- Monitoring of operating conditions
- Basic debugger restrictions (on/off via censorship mode)

5.4 Detailed security information

Details of most of the device security features will be published in a separate document, the *MPC574xx Microcontroller Security Reference Manual*. Please contact your NXP sales representative for details.

Chapter 6
Platform Overview

6.1 Introduction

All MPC574xx family devices supports multiple cores with Power PC Architecture. The MPC5746C Platform overview top level block diagram is shown below.

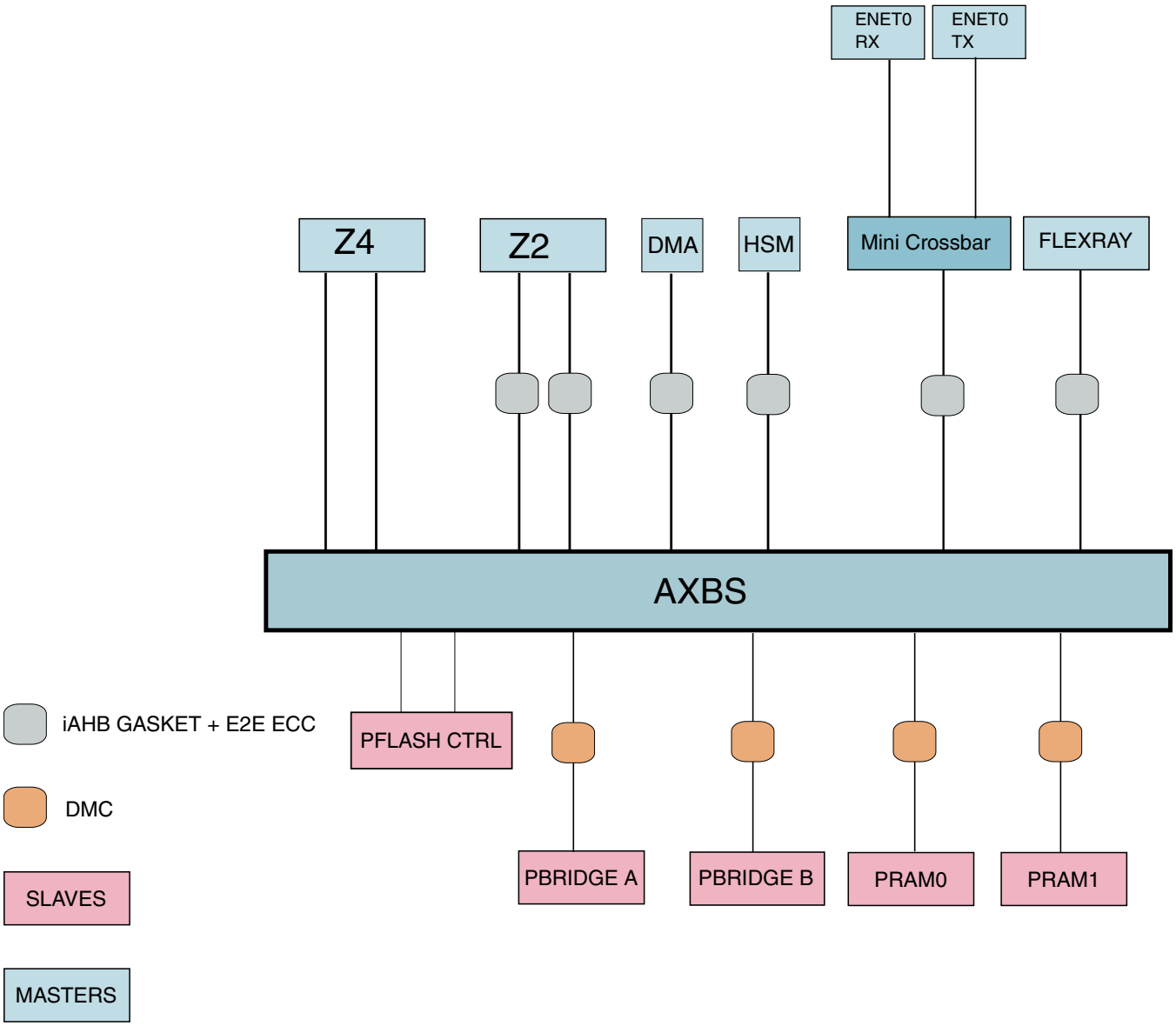


Figure 6-1. Platform Overview

6.2 User/Privilege Access

The non-core masters like ENET, Flexray etc can access the System RAM in either user or privileged mode. The kind of access per master is summarized in the below table.

Table 6-1. User/Privilege Access

Masters	Access
DMA	Master Configurable
HSM	Master Configurable

Table continues on the next page...

Table 6-1. User/Privilege Access (continued)

Masters	Access
ENET 0	Master Configurable
FlexRay	User mode only

NOTE

Bus error response will be generated when read or write access violation occurs for these Masters.

NOTE

Burst accesses via HSM IAHB gasket are not allowed.

Chapter 7
Power Management Overview

7.1 Introduction

The MPC5746C microcontroller's PMC generates, monitors and controls the power supply and related resets in the device. This chapter describes the analog PMC and its integration within the device, specifically detailing how the power supply to various modules is generated. It also explains different supply configurations for the MPC5746C.

The power management infrastructure comprises the following modules:

- Mode Entry Unit (MC_ME)
- LPU Control Unit (LPU_CTL)
- Power Control Unit (MC_PCU)
- Power Management Controller (PMC)

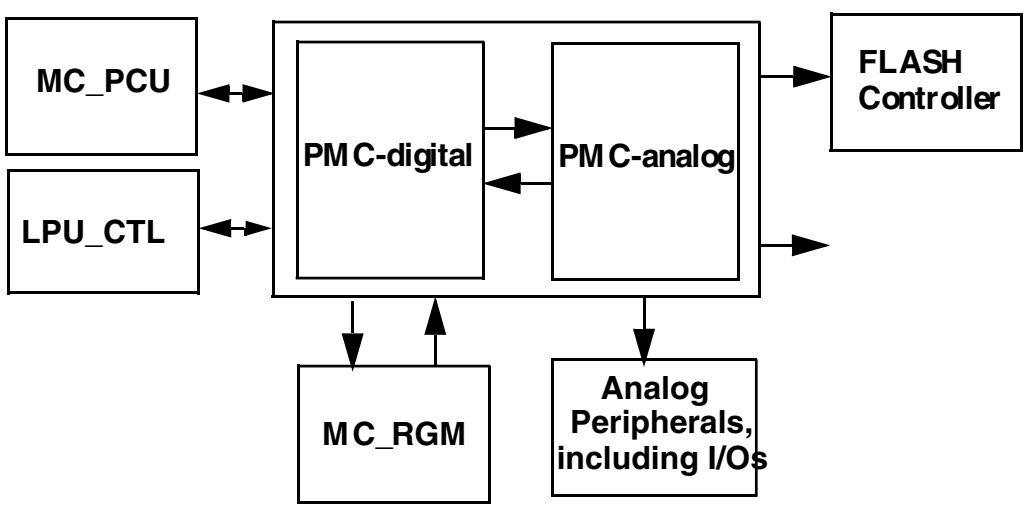


Figure 7-1. Power Management Overview

Finally, the chapter gives a brief introduction about power mode transitions, in conjunction with other sections of this reference manual.

NOTE

The PMC control registers are in the PMC Digital Interface (PMC-Dig) module. See that chapter for details on configuring the PMC.

Table 7-1 defines the abbreviations used throughout this chapter.

Table 7-1. Abbreviations used in this chapter

Abbreviation	Description
Voltage Supplies	
VDD_LV	Core logic low voltage supply
VSS_LV	Core logic ground supply
VDD_HV_A	Main I/O voltage supply
VDD_HV_B	Ethernet voltage supply
VDD_LP_DEC	Low power decoupling capacitor
VDD_HV_FLA	Flash memory high voltage supply
VRC_CTRL	PMC Voltage Regulator Control Output
Internal Regulators	
FLASHREG	Flash regulator. Regulates flash high voltage
FPREG	Full power regulator. Regulates core logic supply during full power operation
FPREG_init	Full power regulator with internal ballast.Regulates core logic supply during full power operation
LPREG	Low power regulator. Active in low power modes
ULPREG	Ultra-low power regulator. Active in STANDBY modes
Other	
LPU	Low Power Unit. An operational mode in which the main processor core and some associated peripherals and logic remain powered while the entire rest of the device is completely power gated off.

See Table 7-2 for links to related information.

Table 7-2. Related information

Link	Description
PMC Digital Interface (PMCDIG) chapter	PMC digital interface module chapter. Contains details on registers used to configure and control the PMC.
UTEST miscellaneous register [UTEST_MISC]	DCF UTEST miscellaneous register. The REG_TYPE bit referenced later in this chapter is used when bypassing the internal core logic power regulator and supplying the core logic voltage from an externally-regulated source.
Modes of Operation	Overview of device operational modes
Mode Entry (MC_ME) Module chapter	Mode Entry module chapter

7.1.1 Features

Features include:

- A linear regulator with an external NPN ballast to generate a regulated 1.25V(nominal) supply (VDD_LV) for the core logic, using external 5V/3.3V supply(VDD_HV_A), with an option to bypass the regulator and provide the core supply externally.
- A linear regulator with an internal ballast(PMOS FET) to generate a regulated 1.25V(nominal) supply (VDD12) for the core logic, using external 5V/3.3V supply(VDDE_A), with an option to bypass the regulator and provide the core supply externally.
- Two low power linear regulators (with internal ballasts) to power the core logic during low power modes (supply VDD_LP_DEC and an internal supply powering the IOP domain).
- Power switches to isolate voltage islands and configure the device in low power modes.
- A linear regulator to generate a regulated 3.3V supply for flash memory (VDD_HV_FLA) when the chip is configured in 5V mode.
- Voltage monitors to ensure the device transitions to a safe state (Power on RESET) when a supply is out of valid range.
- Controls power mode transitions through the interaction with digital interface.

7.2 Power domains

The MPC5746C microcontroller features multiple power domains to provide performance configurations with varying power consumption. The power domains are illustrated in Figure 7-2.

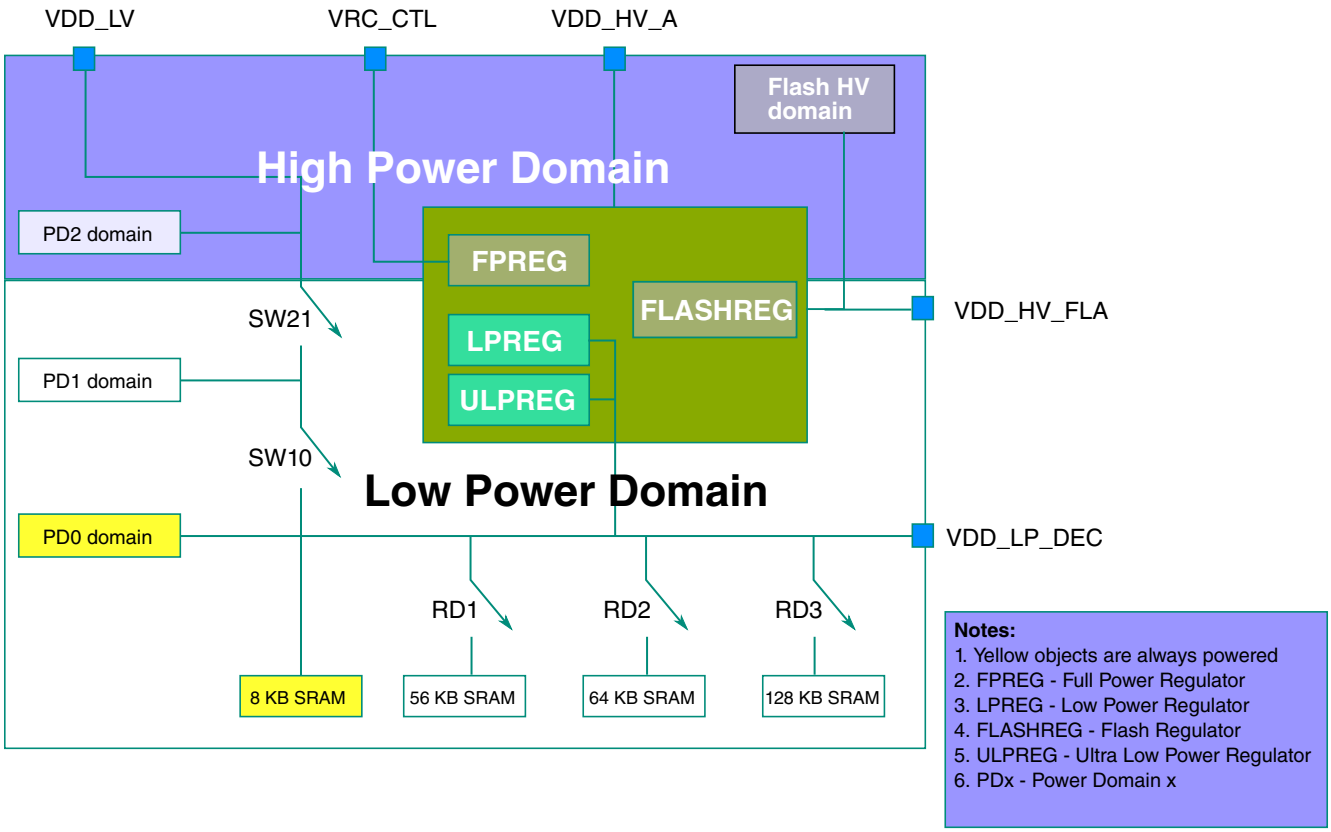


Figure 7-2. Power block diagram

The following sections describe the power domains in detail.

7.2.1 Power Domain0 (PD0)

PD0 is the ‘always ON’ power domain on the device. This domain plays an important role in maintaining device configuration as well as providing basic autonomous functionality with very low power.

7.2.2 Power Domain1 (PD1)

PD1 contains the computational blocks that comprise the LPU domain. This domain is power gated OFF in STANDBY and LPU_STANDBY modes.

7.2.3 Power Domain2 (PD2)

PD2 consists of various high performance blocks. This domain is power gated off in STANDBY mode and all LPU modes.

7.2.4 Flash HV power domain

The Flash HV power domain consists of high voltage devices inside the flash memory module. This domain must be kept powered at all times while the chip is out of 'power on reset'. The domain is supplied by Flash Regulator, as described later in this chapter.

7.2.5 SRAM domains

The MPC5746C microcontroller has a total of 512 KB SRAM, which is divided among the SRAM domains (known as RDx, where x is from 0 to 3) as follows:

- The *RD0* domain contains 8KB SRAM that is always available, and resides in the PD0 power domain.
- The *RD1*, *RD2*, and *RD3* SRAM domains, can be configured to be available during STANDBY.
- The remaining 256 KB SRAM resides in the PD2 power domain. This SRAM is completely powered off in STANDBY and LPU modes, but is always powered on (available) in other modes.

NOTE

All SRAM is available in RUN modes.

7.2.6 SRAM Configuration

256K RAM is available in LPU modes for code execution or data storage. Following guidelines should be considered.

1. RAM Domain Configuration Register (PMCDIG_ RDCR) must be configured to 256K retention mode before enabling LPU subsystem.
2. There is no option to power gate RAM domains in LPU_STOP.
3. RAM Domain Configuration Register (PMCDIG_ RDCR) can be configured to enable desired retention for enhanced power saving in LPU_STANDBY mode.

Table 7-3. RAM retention

RET_RD256	RET_RD128	RET_RD64	Comments
0	0	0	8K RAM retained
0	0	1	64K RAM retained
0	1	x	128K RAM retained
1	x	x	256K RAM retained

7.3 User operating modes, power domains, and clock gating

User operating modes, power domains, and clock gating in the MPC5746C microcontroller are as follows:

- Power domains are logical divisions within the microcontroller that can be power-gated to put the microcontroller in a reduced-power configuration. In other words, a power domain is a group of modules that can all be shutdown by gating the power to that domain.
- A user operating mode is a pre-defined group of power- and/or clock settings that can be applied to the microcontroller while in its normal operational state (DRUN). The settings control power consumption by controlling the availability of device functions.
- Within a given user operating mode, clocks can be configured or gated to achieve desired power targets.

NOTE

- For a full list of the operational modes, see Modes of Operation.

The table below shows the general settings and options available for user operating modes.

Table 7-4. User Operating mode settings and options

Operating Mode	Powered Domains[1],[2]	Chip Features[3],[4],[5]						Clock Sources					Wake up Options
		Z2 core (PD1)	Z4 cores (PD2)	Powered SRAM Domains[6],[7],[8]	Flash (PD2)	LPU Peripherals (PD1)	Main Peripherals (PD2)	16 MHz FIRC	8MHz-40MHz FXOSC	Phase Locked Loop (PLL)	128 kHz SIRC	32 kHz SXOSC	
RUN	PD0 PD1 PD2	OPT	OPT	All	OPT	OPT	OPT	Always ON	OPT	OPT	OPT	OPT	N/A
STOP	PD0 PD1 PD2	CG	CG	All	CG	OPT	OPT	OPT	OPT	CG	OPT	OPT	All peripheral interrupts, refer Wakeup Sources for various modes

Table continues on the next page...

Table 7-4. User Operating mode settings and options (continued)

Operating Mode	Powered Domains[1],[2]	Chip Features[3],[4],[5]						Clock Sources					Wake up Options
		Z2 core (PD1)	Z4 cores (PD2)	Powered SRAM Domains[6],[7],[8]	Flash (PD2)	LPU Peripherals (PD1)	Main Peripherals (PD2)	16 MHz FIRC	8MHz-40MHz FXOSC	Phase Locked Loop (PLL)	128 kHz SIRC	32 kHz SXOSC	
LPU_RUN	PD0 PD1	OPT	Off	RD0 RD1 RD2 RD3	Off	OPT	Off	Always ON	OPT	Off	OPT	OPT	All peripheral interrupts, refer Wakeup Sources for various modes
LPU_STOP	PD0 PD1	CG	Off	RD0 RD1 RD2 RD3	Off	CG	Off	OPT	OPT	Off	OPT	OPT	FLexCAN_0 (PN), no other peripheral interrupts, refer Wakeup Sources for various modes
STANDBY/ LPU_STANDBY	PD0	Off	Off	RD0 RD1[9] RD2 RD3	Off	Off	Off	OPT	OPT	Off	OPT	OPT	WKPU RTC/API Anal Comp, refer Wakeup Sources for various modes

1. *All* means all power domains, i.e., PD0, PD1, and PD2
2. PD0 is always powered during normal device operation
3. *OPT* means optionally configurable to be enabled or disabled
4. *CG* means clock gated, powered but clock stopped
5. *Off* means powered off and clock gated off
6. Depending on the specific device within the microcontroller famly, there are from 1 to 3 on-chip RAM arrays. RAM array 0 contains multiple blocks of RAM of varying sizes and each is defined by an "SRAM domain", RD0-RD3. See SRAM domains for details.
7. On-chip RAM not assigned to an SRAM domain is part of the PD2 power domain and is available only when PD2 is powered.
8. *All* means all RAM including all SRAM domains plus all additional on-chip RAM.
9. RD1, RD2, and RD3 can be independently configured to be available or not during STANDBY/LPU_STANDBY

7.4 Voltage regulators

The MPC5746C microcontroller has a sophisticated Power Management Unit (PMU) comprised of four separate voltage regulators.[1]

7.4.1 FPREG voltage regulator

The FPREG (Full Power Regulator) is a linear regulator that uses an off-chip NPN transistor as a pass device. This regulator is used during internal regulation mode of the microcontroller to generate a regulated 1.25V supply (VDD_LV). The VRC_CTRL pin controls the base of the pass device, while the VDD_LV pins of the device connect to the emitter of the pass device, as shown in diagram below. Note that VDD_HV_BALLAST is just for representation – it can be shorted to the source of VDD_HV_A supply on the application hardware.

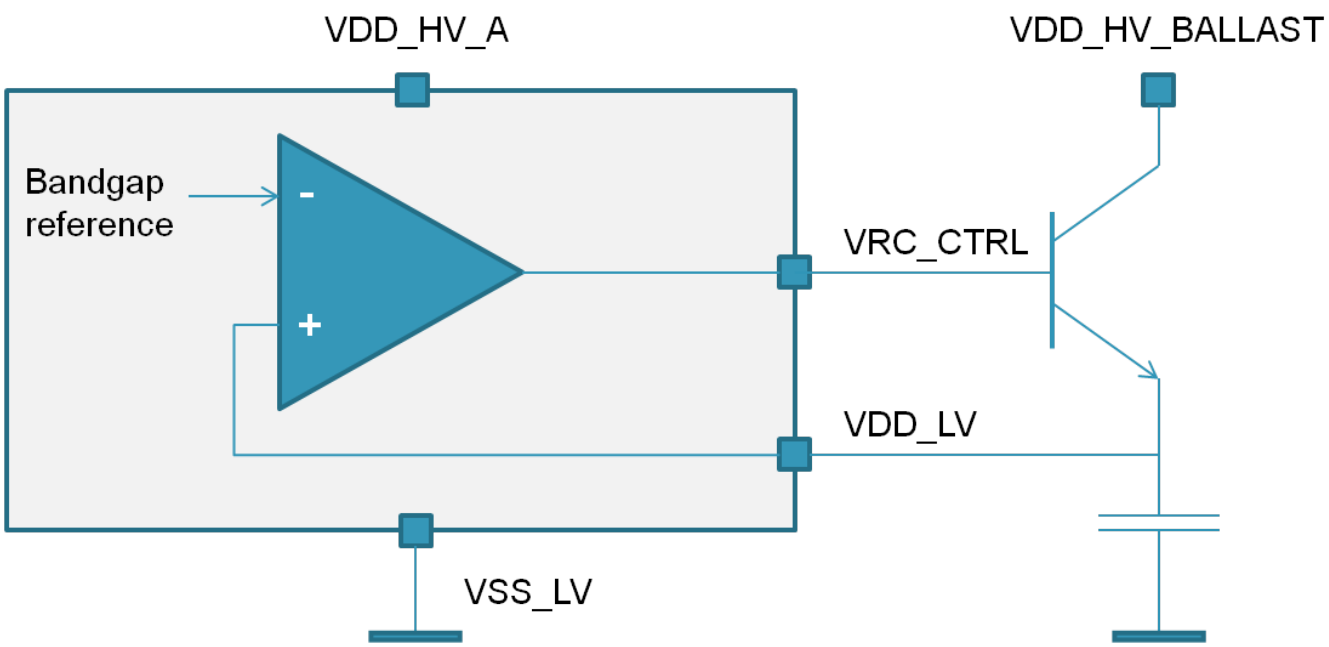


Figure 7-3. FPREG voltage regulator

NOTE

VDD_HV_BALLAST can be an independent supply from VDD_HV_A. Alternatively, these two can be shorted. The minimum voltage for VDD_HV_BALLAST must be 2.5V-10% (2.25V) in order to avoid NPN transistor saturation issues.

1. Refer to datasheet for electrical specifications of these voltage regulators.

Resistor should not be placed at the collector of NPN. Refer Table, Voltage regulator electrical specifications, from DS for related specifications.

If the core logic voltage (VDD_LV) needs to be supplied using an external on board regulator, the FPREG regulator is bypassed by not connecting the NPN pass device (leave the VRC_CTRL pin unconnected).

NOTE

Since the default configuration at boot-up is internal regulation mode, a user option bit UTEST_MISC[REG_TYPE] must be programmed to disable the FPREG regulator when it is not needed.

Disabling of the FPREG regulator is applied during RGM phase 3 when SSCM reads device configuration. Additionally, when using an external VDD_LV supply, the PORST pin must be driven until the supplies are stable. This can be accomplished by driving PORST pin such that when external VDD_LV is not stable, PORST is driven LOW externally, and once VDD_LV is stable, PORST is driven HIGH. Alternatively, another equally robust solution is to connect PORST pin to VDD_HV_A supply. In this latter case, multiple voltage monitor resets may be seen during power-up / power-down in presence of a non-monotonic VDD_LV supply.

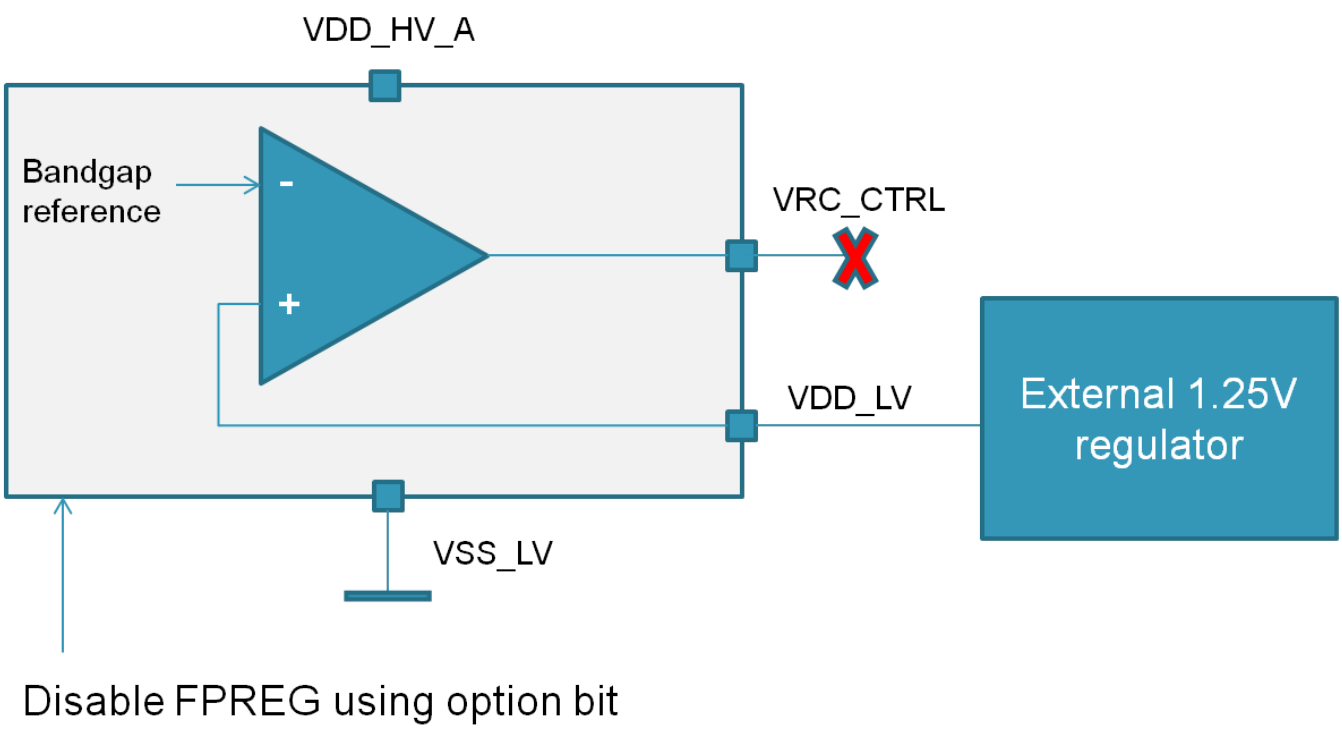


Figure 7-4. Using an externally supplied core logic power supply

NOTE

When using an externally supplied core logic power supply it is acceptable to leave the VRC_CTRL pin floating.

7.4.2 Internal regulation (Internal ballast)

The FPREG has it's ballast sitting on chip. This regulation mode is applicable to only those use cases where the RunIDD numbers are below 160mA, due to thermal limitation of the package.

7.4.3 LPREG voltage regulator

The LPREG (Low Power Regulator) is a linear regulator with an on-chip PMOS pass device. This regulator is active during LPU modes. It powers both PD1 and PD0 domains (and SW10 is closed). During these modes, power domain PD2 is powered off since SW21 is open.

NOTE

- LPREG regulator require an off-chip decoupling capacitor, which must be connected to the VDD_LP_DEC pin. Refer to the data sheet for the electrical specifications of this decoupling capacitor.
- The VDD_LP_DEC supply ramps up sometime after the VDD_LV supply has powered, regardless of whether the FPREG regulator is used.

7.4.4 ULPREG voltage regulator

The ULPREG (Ultra Low Power Regulator) is a very low power linear regulator with an on-chip NMOS pass device. This regulator is active during STANDBY modes and powers the PD0 domain. During this mode, power domains PD2 and PD1 are power gated OFF since SW21 and SW10 are both open.

NOTE

- ULPREG regulator require an off-chip decoupling capacitor, which must be connected to the VDD_LP_DEC pin. Refer to the data sheet for the electrical specifications of this decoupling capacitor.
- The VDD_LP_DEC supply ramps up sometime after the VDD_LV supply has powered, regardless of whether the FPREG regulator is used.

7.4.5 Flash voltage regulator

The Flash regulator is an on-chip linear regulator that generates 3.3 V supply for Flash memory, when the device is configured in 5 V mode (UTEST_MISC[5V_3p3V] in default state). This regulator is bypassed when device is configured in 3.3 V mode, in which VDD_HV_FLA is connected to VDD_HV_A and both are supplied by the same external 3.3V source.

NOTE

The Flash regulator always require an off-chip decoupling capacitor, which must be connected to VDD_HV_FLA pin. Refer to data sheet for electrical specification of this capacitor.

7.4.6 Regulation mode decoding scheme

Internal regulation (internal ballast) Vs internal regulation (external ballast)

INT_BAL_SELECT pin is a dedicated pin , which is used to select regulation mode between internal ballast mode Vs external ballast mode.

INT_BAL_SELECT pin tied to VDD_HV_A on board - Internal regulation (internal ballast).

INT_BAL_SELECT pin tied to ground on board - Internal regulation (external ballast).

Internal regulation Vs external regulation

The ballast is disconnected (un-mounted on board). The customer can apply the core supply externally. Selection of regulation modes between internal and external regulation mode is done by reading a bit in DCF record. In external regulation mode as soon as the DCF is read the internal regulators would be switched OFF.

Table 7-5. Configuration for regulation mode selection

INT_BAL_SELECT (PIN)	REG_TYPE (DCF bit)	FPREG (external ballast regulator)	FPREG_int (internal ballast regulator)	External regulation mode
1	x	No	Yes	No
0	0	No	No	Yes
0	1	Yes	No	No

NOTE

Last row of the table is the default configuration of the device, in which the device works in internal regulation mode with external ballast.

7.4.7 Voltage monitoring

The microcontroller features internal voltage monitoring to ensure supplies are within their correct operating ranges. If any ‘essential’ supply falls outside of its correct operating range, the voltage monitors take the device into a ‘safe, power on reset’ state. The following diagram describes the various voltage monitors and what supplies (and what point on supply) they monitor. Refer to the data sheet for details on electrical specifications for these voltage monitors.

NOTE

- The non essential supplies that are NOT shown in the diagram (VDD_HV_IO_B, VDD_HV_IO_C, VDD_HV_ADC0, VDD_HV_ADC1 and reference voltages VDD_HV_ADC1_REF and VIN1_CMP_REF) below are NOT monitored internally; it is recommended to implement external voltage monitoring on these supplies, if required.
- As shown in the followng figure, hot sense point refers to the point of maximum internal voltage drop due to routing grid. Hot point has the lowest voltage on that supply domain. Conversely, cold sense point refers to the source point for supply, which is the device supply pins.

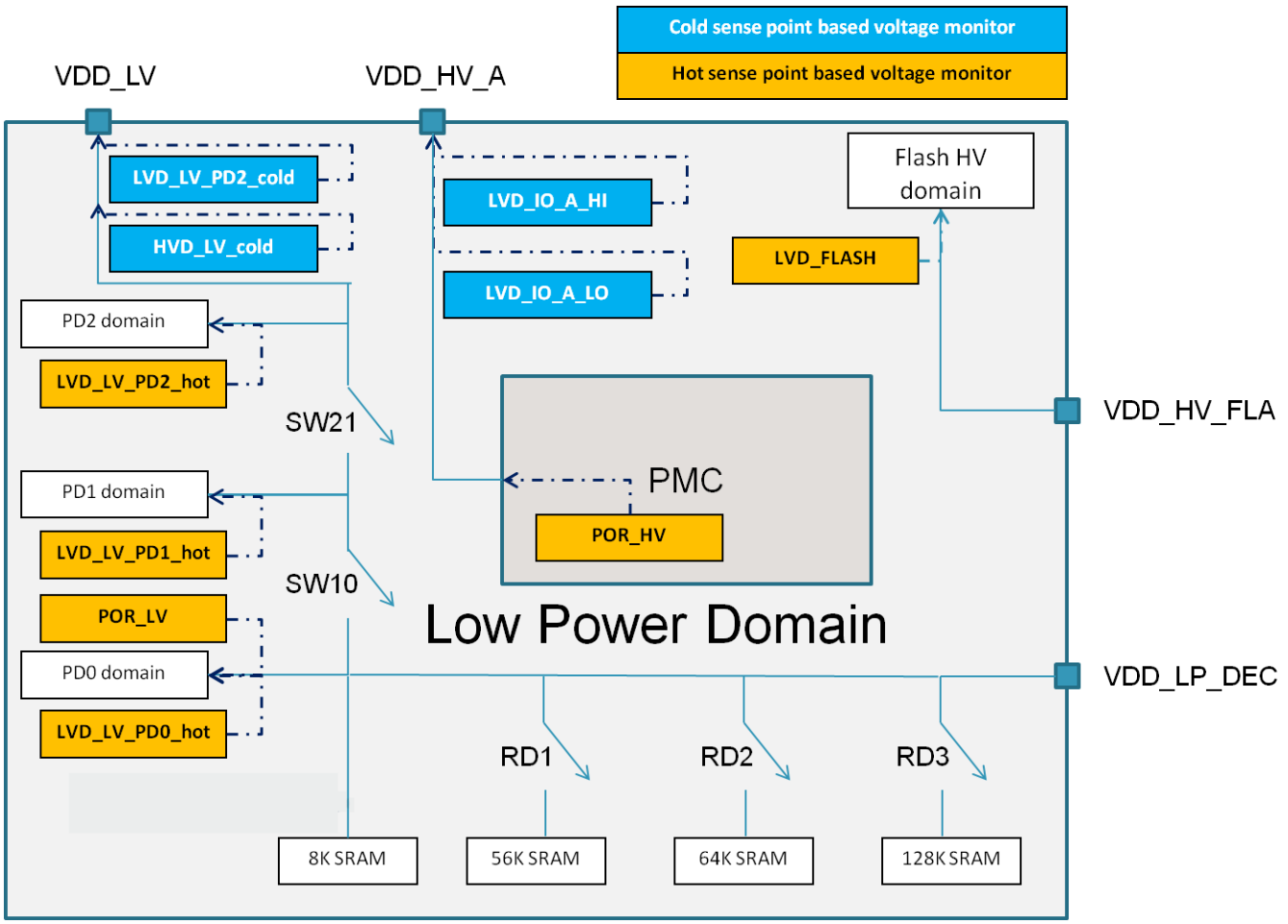


Figure 7-5. Voltage monitoring

The below table provides the complete list of voltage detect signals and their configuration in various modes. For checking there status, refer Supply Gauge Status Register (PMCDIG_SGSR)

Table 7-6. Voltage detect signals

Voltage detect signals	ULP	LP	RUN
POR_LV	ON	ON	ON
POR_HV	ON	ON	ON
LVD_LV_PD0_hot	ON	ON	ON
LVD_LV_PD1_hot	OFF	ON	ON
LVD_LV_PD2_hot	OFF	OFF	ON
LVD_LV_PD2_cold	OFF	OFF	ON
LVD_IO_A_LO	ON	ON	ON
LVD_IO_A_HI	ON	ON	ON
LVD_FLASH	ON	ON	ON
HVD_LV_cold	OFF	OFF	ON

NOTE

When the device is configured in 5V mode, the user should configure LVD_IO_A_HI as a source of functional reset before transitioning to any of the power gated low power modes (STANDBY, LPU etc.). If this configuration is not done, the device may encounter a destructive reset if VDD_HV_IO_A supply drops below 4.5V.

7.4.8 Applications and Low Power Modes

Depending on application profile, a combination of one or more power modes will be useful to optimize power dissipation and overall execution speed. For example, powering off the PD1 and PD2 power domains helps reduce power, particularly at higher temperatures.

NOTE

- When determining the optimum trade-off between fully powered and partially powered modes, consideration should be given to the energy required to recharge the system capacitances (VDD_LV pin decoupling capacitance, see the datasheet for specification).
- It is essential to perform power budgeting calculations to determine which low power mode would be more efficient. In general, the longer the time spent in a low power mode, the more efficient it is to use the advanced power gating features.

7.5 External Regulation

7.5.1 PORST in EXTERNAL REGULATION

It is recommended to drive PORST pin when using external regulation mode, such that when external VDD_LV is not stable, PORST is driven LOW externally, and once VDD_LV is stable, PORST is driven HIGH. Alternatively, another equally robust solution is to connect PORST pin to VDD_HV_A supply. In this latter case, multiple voltage monitor resets may be seen during power-up/power-down in presence of a non-monotonic VDD_LV supply.

During Low Power Modes i.e. all LPU Modes and STANDBY mode, if UTEST_MISC[PORST_MASK] is set to 0 then PORST must be kept high even if VDD_LV is supplied externally or not. Pulling PORST pin low in low power modes will result in chip getting into RESET.

In case UTEST_MISC[PORST_MASK] is set to 1 then PORST pin can be pulled low in low power modes to indicate externally supplied VDD_LV is not present. Also PORST pin can be pulled high if externally supplied VDD_LV is within operating range on low power mode exit.

The device will not exit a low power mode as long as the PORST pin is kept low.

7.5.2 External regulator contol

External regulator control functionality is provided on Pin PG[6]. It indicates if external regulator can be disabled or VDD_LV can be isolated if device is in one of the power gated mode i.e. STANDBY or LPU modes.

- External regulator control functionality on Pin PG[6] can be enabled by configuring UTEST_MISC[EXT_REG_CTRL].
- Polarity of PG[6] can be further controlled by appropriately configuring UTEST_MISC[EXT_REG_POL].
- This indication can be used to control external circuitry and either disable external regulator, or isolate VDD_LV pins from the external regulator as shown in Figure 7-6.

The circuit should be designed in such a way that electrical specs stated in the device datasheet are respected.

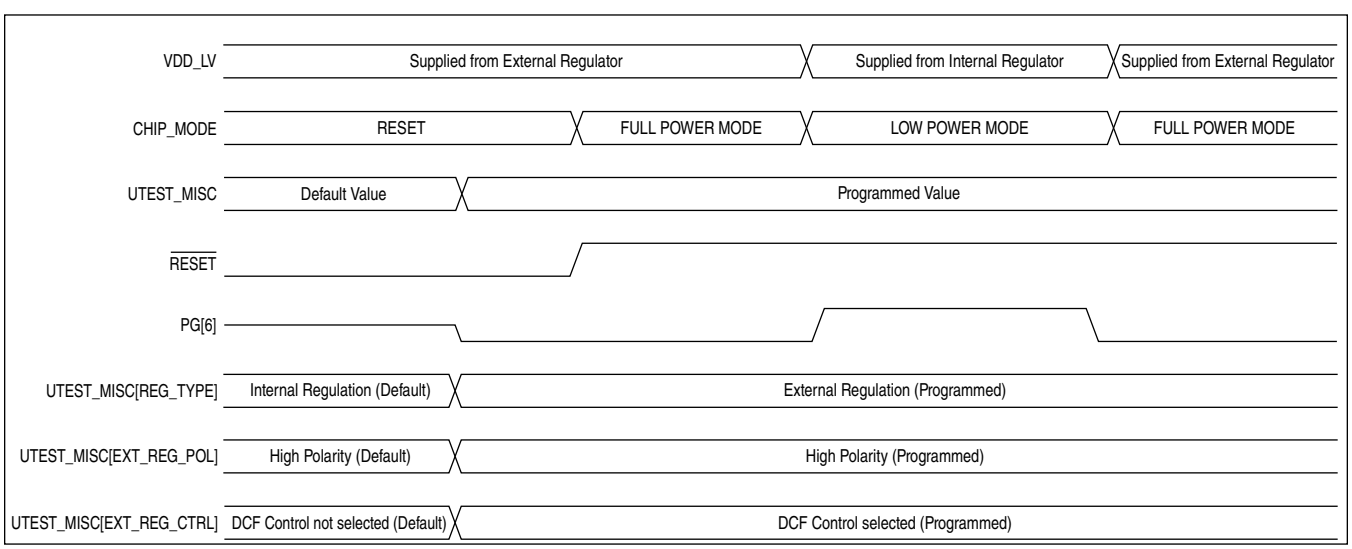


Figure 7-6. External regulator control handshake protocol

NOTE

The circuit is based upon default high polarity of EXT_REG_CTRL control signal. The pass device needs to change (NMOS to PMOS) or an inverter needs to be implemented to modify the required polarity. Refer to UTEST_MISC register for polarity details.

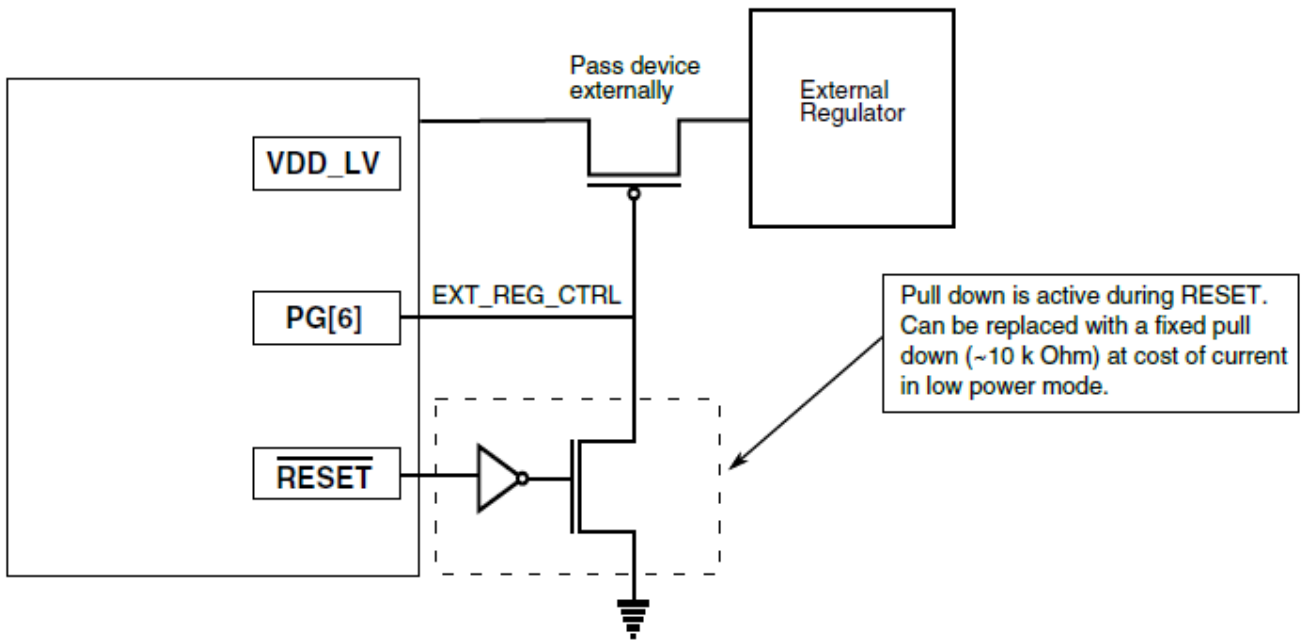


Figure 7-7. External regulator control

Alternatively, if the user is switching off the external regulator in low power mode, it is recommended to pull down the VDD_LV to at least 1.08 V.

In this case, user can program EXT_REG_EXIT_TRIM_WAIT DCF counter to indicate the external regulator stabilization time needed on low power mode exit. See External Regulation Exit Trim Wait (PMCDIG_ERETW) for details.

7.6 Accelerated Low Power Exit

Device supports reduced wakeup time on low power exit LPU or STANDBY, if executing from FLASH. The device can automatically select PLL as system clock source if MC_ME is configured prior to low power mode. This feature allows considerable improvement in low power exit time where HSM is enabled and requires RAM initialization as configured through HSM DCF options.

NOTE

If booting core is z4 and executing from SRAM, this feature cannot be used as Read Wait State needs to be configured to 1 before making read access to SRAM at PLL frequency.

PLL reference clock source is selected by MC_CGM_AC6_SC before low power mode is selected. In case FXOSC is turned off in low power mode then wakeup time will include the stabilization time of FXOSC as configured in FXOSC_CTL[EOCV].

Wakeup time is reduced because of the following enhancements:

- PLL fast lock mechanism is achieved by automatically restoring all the previous programmed PLL Configurations on low power mode exit thus saving calibration time and achieving lock faster.

NOTE

Only Last Programmed PLL configuration gets restored which was used to attain PLL Lock

.

- Automatic selection of PLL as System clock source on low power exits, thus flash and core initialization and mode transition to desired RUN can happen at faster clock frequency than FIRC.

Required Software Configurations

- ME_DRUN_MC[SYSCLK] = 0x0002 and ME_DRUN_MC[PLLON] is SET.
- ME_DRUN_MC[FXOSCON] should also be set before entering low power mode,if FXOSC is selected as the reference clock or FXOSC is available while exiting low power mode. See DRUN Mode. GPR_CTL[0] should be programmed to 1 before entering low power mode, if HSM is enabled.
- ME_DRUN_MC[SIRCON] is recommended to be set before entering low power mode,if SIRC is available while exiting low power mode.
- ME_DRUN_MC[SXOSCON] is recommended to be set before entering low power mode,if SXOSC is available while exiting low power mode. See DRUN Mode.
- PLL_DIG configured for bypass of calibration to achieve fast PLL lock. Refer PLL Wakeup from low power mode for further details.

Chapter 8
Low Power Subsystem (LPU)

8.1 Introduction

The low-power unit (LPU) is a subsystem that enables the chip to perform simple tasks with minimal power consumption. When LPU subsystem is active majority of the chip can remain power gated, thus achieving the minimal power consumption. Power Domain 1 (PD1) and Power Domain 0 (PD0) are together referred to as the LPU domain. Power Domain2 is power gated when the LPU subsystem is active. Refer to Modes of Operation for more information.

Operating modes in LPU subsystem are referred to as LPU modes. Z2 core executes from the 256 KB system RAM in LPU RUN modes. Maximum operating frequency in LPU modes is 40 MHz. Tasks performed in LPU modes can include:

- Reacting to incoming messages on the FlexCAN or LINFlexD buses.
- Performing ADC conversions and processing the results.

The chip can enter into LPU mode from any of the RUN modes by configuring the MC_ME register definition and the LPU Controller (LPU_CTL). Refer to Transition to LPU mode. LPU_CTL controls all the mode transitions and clock source configurations in the LPU subsystem independent of the mode entry module.

8.2 Block diagram

The e200z2 core has access to all the peripherals in the LPU subsystem and executes from the 256 KB RAM.

Modules in power domain 0 (PD0) are always powered. Modules in power domain1 (PD1) will be power gated in LPU_STANDBY mode. Refer to Modes of Operation for more details.

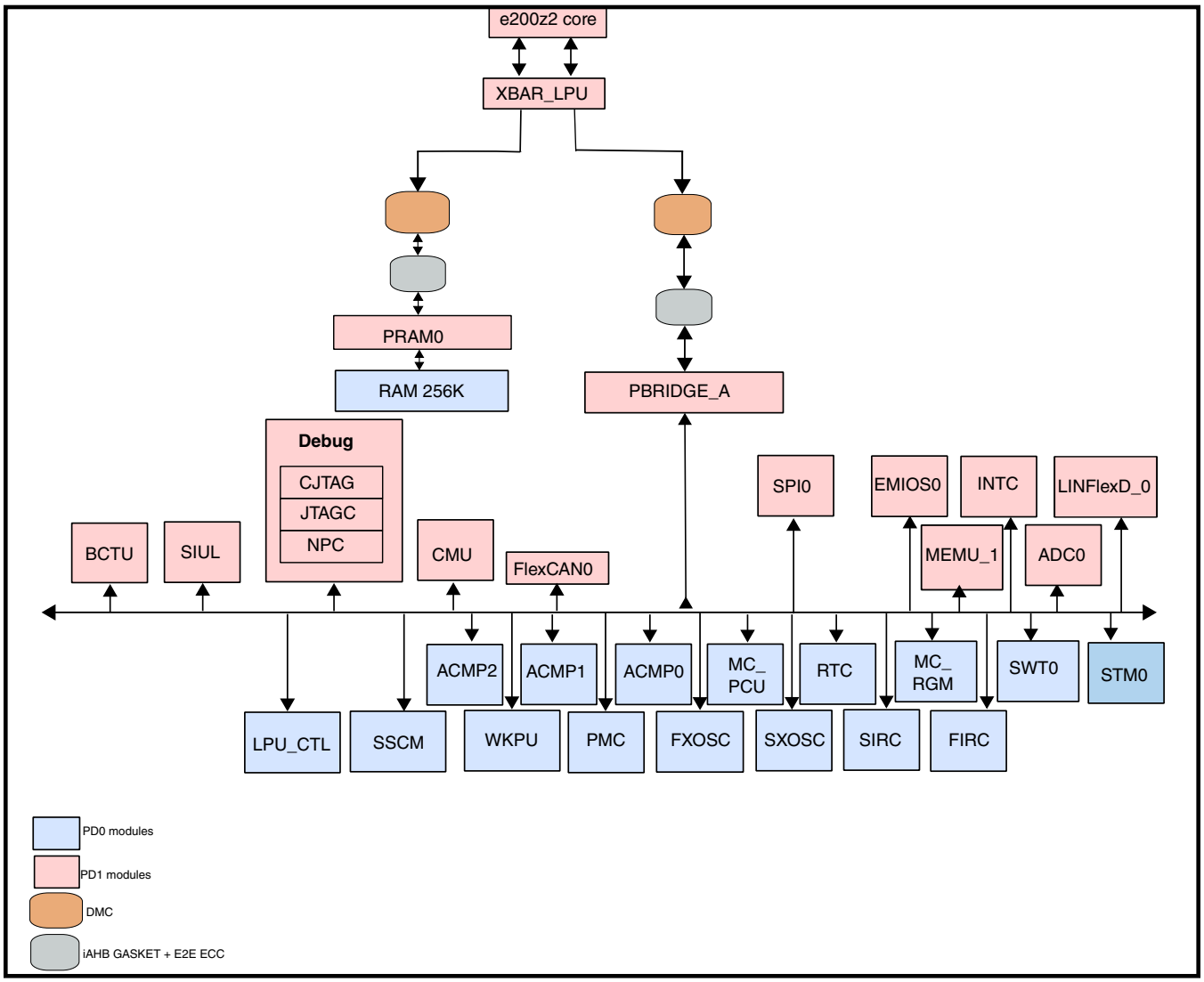


Figure 8-1. LPU subsystem diagram

8.3 Features

- The LPU subsystem can enter into three low-power modes:
  - LPU_SLEEP
  - LPU_STOP
  - LPU_STANDBY
- Nexus class one seamlessly through RUN and LPU mode
- Debug through LPU_STOP and LPU_STANDBY is supported
- Full interrupt support from LPU domain peripherals
- Clock sources available:
  - FIRC
  - FXOSC

    - SIRC
    - SXOSC
- A divided FIRC or FXOSC are available as a system clock source.
- Clock monitoring unit available for measurement of frequency for all clock sources and frequency range check for FXOSC.
- The MEMU module captures faults from the following four sources and raises an interrupt or NMI to the z2 core:
    - AHB_E2E_ECC error from PBRIDGE A
    - z2 intstruction bus error
    - z2 data bus error
    - PRAM_E2E_ECC error from PRAM_CTL0
- Pretended Networking supported on FlexCAN0. It is also possible to wake up from LPU_STOP to LPU_RUN with only the protocol engine clock of FlexCAN0 working when a valid match is found on received message buffer. For details, refer FlexCAN configuration.
- Configurable FlexCAN0 wakeup events:
    - Wakeup on “bus off”
    - Matching message: Receives a pre-configured message
    - Matching message and data: Pre-configured message and RX data matches value
    - Matching range of messages: RX a range of pre-configured messages
    - Matching message range and data
    - Matching data received n times
    - Pre-configured data missing over specific time
    - Pre-configured network management message missing over a specific time
    - Payload length error if a pre-configured message

- Configurable FlexCAN0 Transmit Messages
    - Transmit network management message
    - Transmit pre-configured static message

NOTE

LPU_STANDBY mode is the mode when power domain 1 is power-gated. This mode is entered from LPU_RUN mode. If the chip enters power-gated mode from normal RUN mode, it is referred as STANDBY mode. The only difference between the two modes is how the clock sources are configured. Select the clock sources to be on or off by configuring:

- ME_STANDBY0_MC for STANDBY mode
- LPU_STANDBY_CF for LPU_STANDBY mode

8.4 Modes of operation

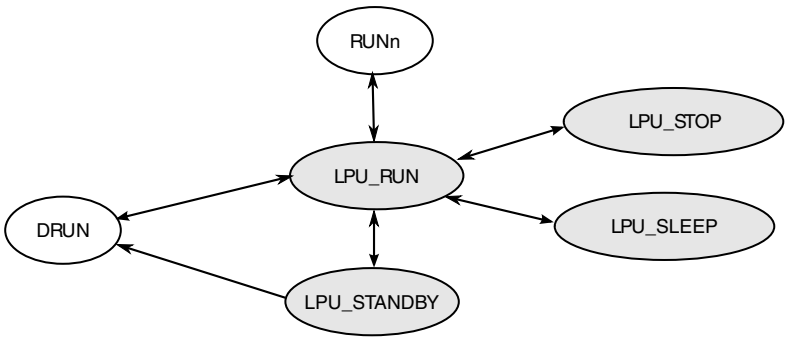


Figure 8-2. LPU modes

- **LPU_RUN**
  - All modules in the LPU domain have their clocks enabled.
  - The chip can enter into LPU_RUN mode under the following conditions:
    - From any of the RUN modes by configuring the ME_MCTL and LPU_MCTL registers.
    - On exit from LPU_STOP,LPU_SLEEP
    - On exit from LPU_STANDBY, if LPU_MCTL[DIRECT_DRUN] is not set.
  - The chip can exit to DRUN, LPU_STOP, or LPU_STANDBY only via configuring the LPU_MCTL register.
  - FIRC is always ON during either LPU_RUN or LPU_SLEEP mode. All other clock sources can be disabled. LPU domain modules can be configured via LPU_MDIS to be clock gated. To change clock source configuration or clock gate modules, an LPU_RUN-to-LPU_RUN mode transition is required.
- **LPU_SLEEP**
  - In this mode, the z2 core enters low-power state and its clock is gated. All other modules in the LPU domain have their clocks enabled by default and can be disabled by configuring LPU_MDIS.
  - The device can enter into LPU_SLEEP mode from LPU_RUN mode.
  - The device can exit to LPU_RUN mode on any peripheral interrupt or external interrupt, or WKPU source.
  - All the LPU peripheral interrupts can cause exit from LPU SLEEP.
- **LPU_STOP**
  - In this mode, the z2 core enters the low-power state and its clock is gated. FLexCAN pretended network feature is available but all peripherals are stopped, clocks are gated to all modules in LPU domain. Clock sources are enabled or disabled by configuring the LPU_STOP_CF register.

- The chip can enter LPU_STOP mode from LPU_RUN mode.
- The chip can exit to LPU_RUN mode when a wakeup event comes from WKPU, SWT timeout, RTC/API, or FlexCAN pretended network interrupt.

NOTE

There should be no pending interrupts or outstanding transactions as it may not allow entry in LPU_STOP. To centrally ensure this, all modules can be disabled by configuring LPU_MDIS in LPU_RUN to LPU_RUN mode transition followed by LPU_STOP mode entry. To bypass STOP acknowledge handshake with LPU domain modules, refer GPR_CTL[LPU_STOP_ACK_BYPASS].

- **LPU_STANDBY**
  - This mode is entered when the chip enters STANDBY mode from LPU_RUN mode.
  - All modules in the PD1 domain are power gated.
  - Device can exit to LPU_RUN mode or DRUN mode when a wakeup event comes from the WKPU, SWT timeout interrupt, or RTC/API timeout interrupt. An Analog comparator can also provide wakeup functionally, if enabled.To exit to DRUN mode, LPU_RUN_CF[DIRECT_DRUN] should be set to 1 before requesting for LPU_STANDBY mode transition.

The following guidelines should be observed when preparing to enter an LPU mode before initiating further low power mode request in LPU:

- LPU_SR[PD2_PWR_STAT] should be polled to be 0 to ensure PD2 is power gated.
- FIRC_CTL[FIRCDIV] should be configured to 0x0 before initiating LPU_STANDBY mode request. It should be reconfigured appropriately on exit to LPU_RUN or DRUN from LPU_STANDBY exit
- PMC_RDCR[PAD_KEEP_EN] should be configured appropriately.
- PMC_RDCR[RD*_RETN] should be configured appropriately

8.5 Transition to LPU mode

The following flowchart shows the hardware state transition from user RUNx/DRUN mode to LPU subsystem RUN mode, that is, LPU_RUN.

Care should be taken that there are no active ongoing transactions on power domain 2 peripherals. Clocks to the processor are gated after the acknowledgement of the low-power stop request from the MC_ME.

The LPU subsystem is enabled after gating the main XBAR clock, while the power gating mechanism is ongoing. Enabling the LPU subsystem before power gating PD2 will allow the z2 core to be available as early as possible to service power domain 1 peripherals, if required.

Clock source enabling and disabling is controlled by the LPU_RUN_CF register as soon as TARGET LPU_RUN mode is requested.

NOTE

The device can enter LPU_STOP or LPU_STANDBY mode from LPU_RUN mode only. The following mode transitions are illegal and if attempted will result the chip to go into STANDBY mode:

- Any RUN/DRUN to LPU_STOP
- Any RUN/DRUN to LPU_STANDBY

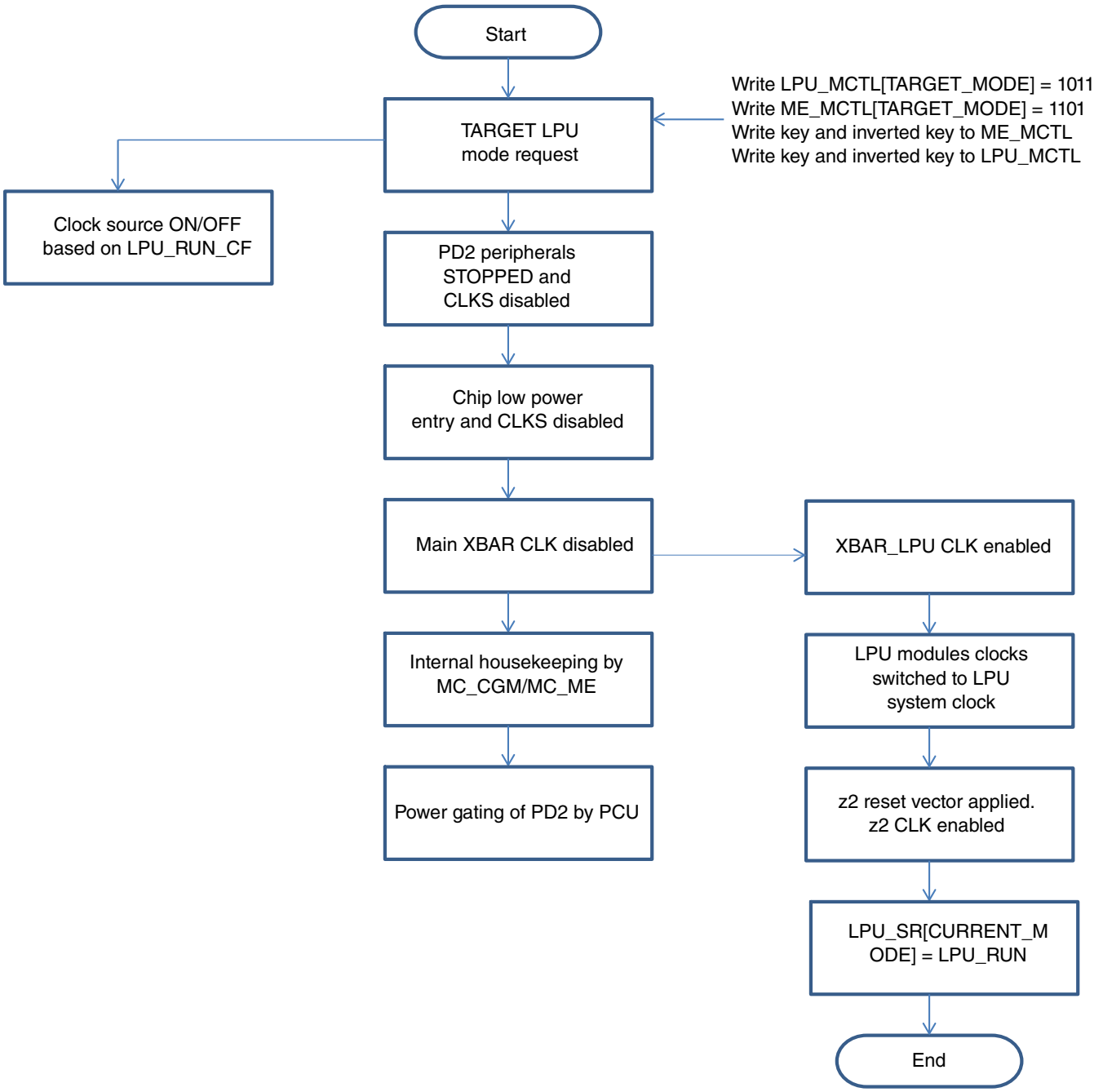


Figure 8-3. Mode transition to LPU mode

NOTE

The transition to LPU mode will not complete if any of the cores are in debug mode.

The following basic code snippet demonstrates how the registers and bit fields are programmed to transition from DRUN to LPU_RUN mode. LPU entry can be carried out on any of the cores: z4 or z2.

```
void LPU_RUN_init ()
```

Transition to LPU mode

```
{
LPU.LPU_RUN_CF.B.CRSxON = 1;           /* Enabled clock sources for LPU_RUN mode. */

/* LPU_RUN init */
LPU.LPU_MCTL.B.TARGET_MODE = 0b1011;   /* Request for LPU_RUN mode */
LPU.LPU_MCTL.B.KEY = 0x5AF0;           /* Write the key to initiate mode change. */
LPU.LPU_MCTL.B.KEY = 0xA50F;           /* Write the inverted key to finalise the */
                                       /*  mode change. */

/* Mode Entry - Standby0 mode init */
/* In order to enter into LPU_RUN mode, Standby0 must be */
/* initialized to disable non-required functionality. */
MC_ME.MC_ME_MCTL.B.TARGET_MODE = 0b1101;   /* Configure ME for standby0 entry. */
MC_ME.MC_ME_MCTL.B.KEY = 0x5AF0;           /* Write the key to initiate mode change. */
MC_ME.MC_ME_MCTL.B.KEY = 0xA50F;          /* Write the inverted key to finalise the mode
change. */
                                                                    /* Loop here until the
Standby0 mode */
                                                                   /*  transition has
occurred by reading */
                                                                 /*  back the current
operating mode. */

/* Loop here until the LPU mode transition has occurred */
/* and also read back the current operating mode. */
while((LPU.LPU_SR.B.MTRANS != 1) && (LPU.LPU_SR.B.CURRENT_MODE = 0b1011));
}
```

The following guidelines should be observed when preparing to enter an LPU mode:

- Ensure steps have been taken to initialize the appropriate core via the Mode Entry registers MC_ME_CADRR*n* and MC_ME_CCTL*n*.
- LPU clock source configuration must be done prior to an LPU mode transition request because the new configuration is selected only after a valid mode transition.
- Prepare to enter LPU modes by writing appropriate LPU target mode and keys to the LPU Mode Control Register, LPU_MCTL.
- To complete the transition, enable STANDBY mode via the Mode Entry registers, which provides the necessary preparation of disabling functionality not required within the LPU modes.Please ensure all software guidelines mentioned for STANDBY mode in MC_ME chapter are followed here.
- LPU mode transition status, LPU_SR[MTRANS], is asserted only after the mode entry target mode is configured to be Standby0.
- Complete the transition by reading the LPU status register, LPU_SR, for transition complete and current mode as expected.
- Register protection configuration should follow the guideline mentioned in section: Register protection configuration

NOTE

SWT_0 enabling or disabling is prohibited in LPU modes. It should be configured prior to entry in LPU.

NOTE

After initiating any mode transition, the software should poll for LPU_SR[MTRANS] before initiating any other mode transition.

For the DRUN -> (LPU modes) -> DRUN transition, any one of the following two conditions should be satisfied by the software:

- If FXOSC is to be used in LPU modes as LPU_SYS_CLK, then FXOSC should be kept ON in DRUN before/after LPU mode is entered/exit.
- If LPU RUN is entered with FXOSC off in DRUN mode, then LPU_SYS_CLK should be made as IRC before entering into LPU STANDBY.

8.6 Exit from LPU Modes

The following flowchart shows the hardware state transition from user LPU_RUN to DRUN mode.

Exit from LPU_RUN to DRUN

This can be initiated by software by configuring LPU_MCTL. LPU_CTL requests PMCDIG to initiate power up of Power Domain 2. Glitch free clock switching occurs from LPU_SYS_CLK to CGM_CLK on LPU mode exit. LPU peripherals will be clock gated. Care should be taken that there are no active ongoing transactions on LPU peripherals before requesting exit to DRUN mode. Clocks to the processor are gated after the acknowledgement that Power Domain 2 is powered. Power Domain 2 goes through reset sequence and z2 core is enabled if it was configured via ME_CADDR and ME_CCTL prior to entry in LPU modes.

Clock source enabling and disabling and mode transition controls are now transferred back to ME from LPU_CTL.

Exit from LPU_STANDBY to DRUN

Any wakeup source can take the chip back to DRUN mode, if LPU_RUN_CF[DIRECT_DRUN] was configured to 1 before LPU_STANDBY mode transition request was initiated. Exit sequence is identical to exit from DRUN except that LPU peripherals and processors were already power gated in LPU_STANDBY, so no housekeeping is required by software

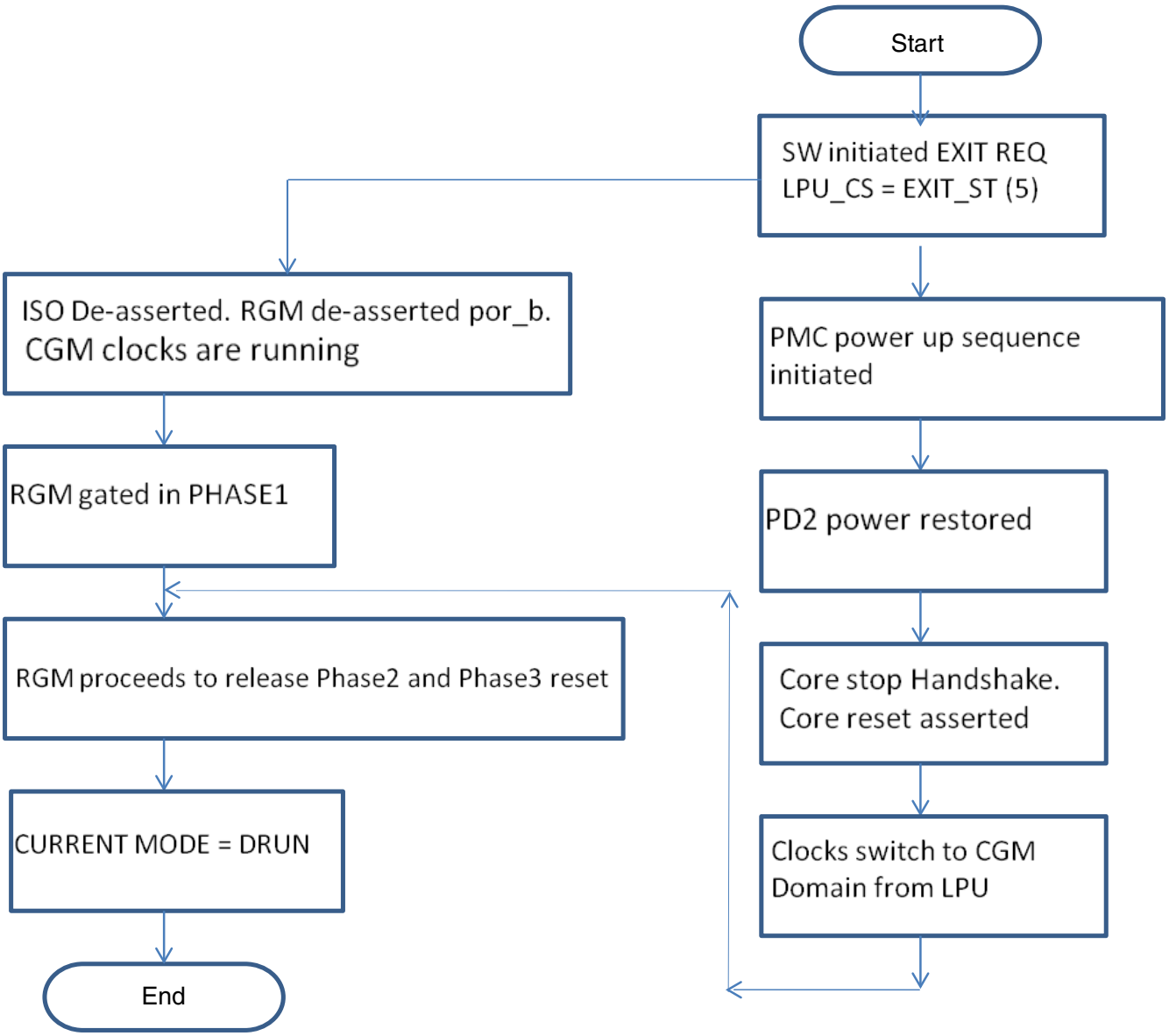


Figure 8-4. Exit from LPU_RUN to DRUN

8.7 Clocking

8.7.1 LPU System Clock Generation

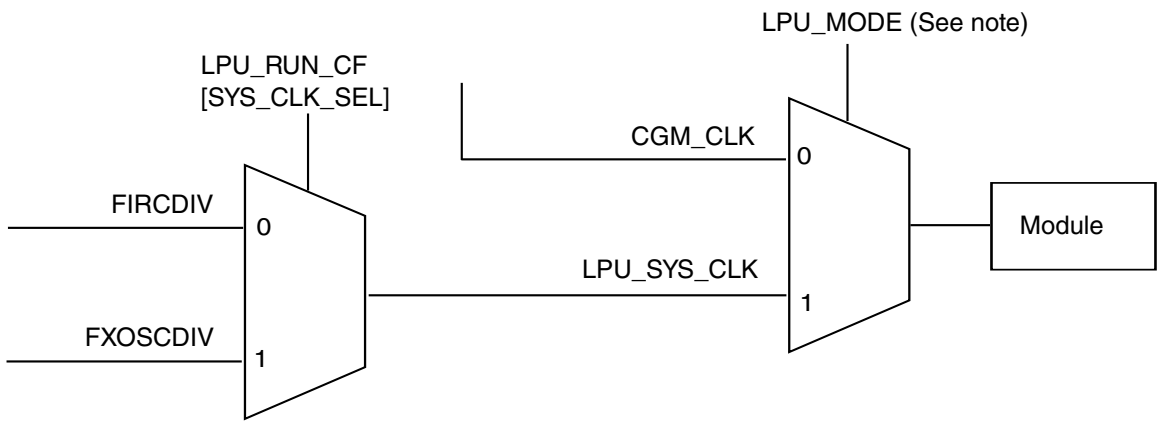
The two primary clock sources, FIRC and FXOSC, are the source of the system clock network in the LPU domain (see Clocking chapter for details). The 16 MHz FIRC clock is the default clock source in LPU_RUN mode. The frequency of the LPU system clock

can be scaled down by configuring the FIRC and FXOSC dividers via the FIRC_CTL and FXOSC_CTL registers, respectively. SIRC and SXOSC are also available for timers, for example, SWT and RTC (see SWT and RTC).

In LPU modes, the LPU system clock is the multiplexed output of either the FIRC divider or the FXOSC divider as shown below. The maximum operating frequency is 40 MHz.

All the module clocks switch from MC_CGM (CGM_CLK) to LPU_SYS_CLK. For example, the z2 clock is sourced from the S80 group, but in LPU mode it will be sourced from the LPU_SYS_CLK. There is a multiplexer between CGM_CLK and LPU_SYS_CLK for each clock frequency node in frequency groups, that is, F160, S160, S80, F80, F40, S40, FS80, and F20. Refer to Clock Generation Module (MC_CGM) for clocking details.

All the module clocks will be gated for period less than or equal to 2 clock cycle of current clock plus 2 clock cycle of next clock period to allow glitch free switching of clocks to LPU_SYS_CLK.



Note: LPU_MODE implies any of the three LPU modes - LPU_RUN, LPU_STOP, LPU_SLEEP, and LPU_STANDBY. There is no software control for this mux. Switching is handled during RUN to LPU_RUN transition protocol.

Figure 8-5. LPU clocking

NOTE

If the CGM mapped peripherals registers of module FIRC, FXOSC, SIRC, SXOSC, CMU are protected under Register protection scheme before transitioning to LPU mode then the protected registers will not be accessible during LPU mode for further write operation. Protected register would be available for read operation. Hence if any register is required to be modified during LPU mode then software should ensure that these registers should remain unlocked under register protection scheme.

NOTE

FIRC, FXOSC, SIRC, SXOSC, CMU registers cannot be locked under register protection scheme during LPU mode.

8.7.2 FlexCAN0 Clocking

CAN0 Clocking

During an LPU mode, working clock of all modules except CAN0 will be the same as the LPU system clock.

Device can be configured to support seamless operation of CAN0 during LPU_RUN mode entry and exit. This can be ensured by providing continuously running clock through an independent mux. This configuration should be done before CAN0 is enabled in device DRUN/RUNx modes. CAN0 CHI and PE interfaces would be synchronously operating with frequency ratio 1:1 and corresponding CAN network configurations should be done based on max CHI frequency equal to FXOSC frequency.

- MC_CGM_AC9_SC should be configured to select FXOSC. Refer FlexCAN clocking
- LPU_RUN_CF[CAN0_CONT] should be SET to ensure seamless operation on LPU_RUN to DRUN. Refer LPU Memory map and register definition

8.7.3 LinFlexD Clocking

LINFlexD[0] requires 2 clocks for operation : BUS_CLK and BAUD_CLK. Frequency of these clocks should satisfy the equation freq(BAUD_CLK)/3 is less than freq(BUS_CLK) and freq(BUS_CLK) is less than BAUD_CLK.

- **BAUD CLK:** Baud clock is generated from F40 node It is recommended to select FXOSC as source of baud clock. The baud clock may disappear for few cycles while switching from CGM_CLK to LPU_SYS_CLK.
- **BUS CLK:** While entering LPU modes, F20 node which is the BUS_CLK of LINFlexD[0] is same as LPU_SYS_CLK. A divider is automatically enabled to scale F20 node by 2 in LPU modes. Refer LINFlexD clocking

8.7.4 SPI_0 Clocking

SPI requires 2 clocks for operation : BUS_CLK and BAUD_CLK. BAUD_CLK is absolutely asynchronous to BUS_CLK. Baud clock is generated from a mux which is controlled by MC_CGM_AC8_SC. It is recommended to select FXOSC as source of

baud clock. If F40 is selected as source of baud clock than this clock may disappear for few cycles while switching from CGM_CLK to LPU_SYS_CLK. Refer DSPI/SPI clocking

8.7.5 LPU clock source control

For all LPU modes, clock source enabling and disabling is controlled by the LPU mode configuration registers.

FIRC cannot be disabled in LPU_RUN mode or LPU_SLEEP mode. All the module clocks, that is, clocks that are operating from the LPU system clock, will be gated in the LPU_STOP mode and the LPU_STANDBY mode. For more information, see the Table 7-4.

All clock sources can be disabled in any of the low-power modes, that is, LPU_STOP and LPU_STANDBY.

8.8 Reset

Reset generation and control is supported through the MC_RGM module. Both destructive and functional resets are supported.

The only reactions possible via functional reset sources is Long Reset or Short Reset. It is not possible to generate a SAFE mode request or interrupt via the MC_RGM while in LPU mode.

On reset, the chip will exit to DRUN mode from any of the LPU modes.

NOTE

During Low Power Modes i.e. all LPU Modes and STANDBY in external regulation mode PORST must be kept high even if VDD_LV is supplied externally or not. Pulling PORST pin low in case VDD_LV is not driven externally in low power modes will result in chip getting into RESET.

Following are the active reset sources in the LPU mode.

Table 8-1. Destructive reset sources

Reset Source Mapping	Destructive Reset Source	Reactions Possible	Bidirectional External reset generation/Software control
7	SWT0 (flash initialization error)	Destructive	Yes/Read_only

Table 8-2. Functional reset sources

Reset Source Mapping	Functional Reset Source	Reactions Possible	Bidirectional External reset generation/Software control
4	NMI from wakeup unit	Long/Short	No/Read_Write
5	JTAG Reset	Long/Short	No/Read_Write
7	FXOSC frequency out of range from CMU	Long/Short	No/Read_Write
12	z2 Debug Reset	Long/Short	No/Read_Write
13	LVD_IO_A_HI Reset	Long/Short	No/Read_Write

NOTE

If the application software is required to trigger a software reset in LPU_RUN this can be achieved via z2 core register DBCR0[RST]. This reset will be flagged at RGM_FES[F_Z2_DBG].

The LPU_CTL can enter, gate, or control the reset process as listed in the following table together with their type of contribution.

Table 8-3. Reset Process Modules

Module	Contribution Type		
	event	gating	control
LPU_CTL	Yes	Yes	Yes

The types of contribution are described in the following table.

Table 8-4. Module contribution type

Contribution Type	Description
event	The module may generate events which may trigger the entry into a reset state, if the event configuration in the MC_RGM allows this.
gating	The module can prevent the exit of a reset state if it has not reached a certain internal state.
control	The module controls the sequence of the reset process.

8.9 Interrupts

Interrupts from modules in the LPU domain are supported. Interrupt mapping is consistent across full modes to LPU modes. Refer to INTC interrupt sources for details.

NOTE

If any interrupt is pending before requesting mode transition to LPU_STOP and LPU_STANDBY, the mode transition request would be ignored.

8.10 Boot

On entry into LPU_RUN mode, the z2 is reset and starts executing from SRAM0 at 0x4000_0000 on reset de-assertion. On LPU_STANDBY exit to LPU_RUN mode, the z2 starts executing from SRAM0 at 0x4000_0000. The device can boot from any core on exit from LPU_RUN/LPU_STANDBY mode.

Upon LPU_STANDBY exit to DRUN mode, which can happen if the exit occurs due to a reset event or LPU_RUN_CF[DIRECT_DRUN] bit is set before entering in LPU_STANDBY, the z2 will start executing from the address configured in MC_ME[CADDR3] if it is enabled in MC_ME[CCTL3]. For more information see, Table 13-18.

NOTE

Configure MC_ME[CADDR*n*] and MC_ME[CCTL*n*] prior to entering LPU_RUN mode.

8.11 LPU Controller (LPU_CTL)

The LPU controller module controls the mode transition sequences for LPU modes. It also contains configuration, control, and status registers accessible to the application. The LPU controller takes ownership to control the mode from the MC_ME module whenever the chip enters LPU_RUN mode. The LPU controller generates the necessary control signals during various LPU modes for the following:

- clock gating and clock source control
- reset control
- debug handshake across RUN to LPU_RUN and within LPU_MODES

8.11.1 Memory map and register definition

LPU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	LPU Mode Control Register (LPU_MCTL)	32	R/W	3000_A50Fh	8.11.2/172
4	LPU Run Configuration Register (LPU_RUN_CF)	32	R/W	0000_0001h	8.11.3/174
8	LPU Stop Configuration Register (LPU_STOP_CF)	32	R/W	0000_0001h	8.11.4/176
C	LPU Standby Configuration Register (LPU_STANDBY_CF)	32	R/W	0000_0001h	8.11.5/177
10	LPU Status Register (LPU_SR)	32	R	0001_8003h	8.11.6/178
14	LPU Interrupt control register (LPU_ICR)	32	w1c	0000_0000h	8.11.7/181
18	Module Disable (LPU_MDIS)	32	R/W	0000_0000h	8.11.8/183

8.11.2 LPU Mode Control Register (LPU_MCTL)

This register is used to trigger software-controlled LPU mode changes. Software can configure LPU_MCTL[TARGET_MODE] to DRUN, LPU_STOP, or LPU_STANDBY only when the current mode is LPU_RUN.

Byte and half-word write accesses are not allowed to this register because a predefined key is required to change its value.

NOTE

Further low power mode transitions i.e. LPU_STANDBY and LPU_STOP should not be requested if LPU_SR[PD2_PWR_STAT] is 1. Software should wait for PD2 to be power gated before requesting further low power modes.

NOTE

Software should always check PMCDIG_MCR[LP_ST_CHG_IN_PRGRS] is 0 before a new mode transition request is initiated.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TARGET_MODE				0												KEY															
W																																
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

LPU_MCTL field descriptions

Field	Description
0–3 TARGET_MODE	Target Chip Mode This field provides the target mode to be entered by software. The mechanism to enter into any mode by software requires two write operations:

Table continues on the next page...

LPU_MCTL field descriptions (continued)

Field	Description
	1. Write key. 2. Write inverted key. **NOTE:** LPU_SLEEP is entered by configuration of LPU_RUN_CF[LPU_SLEEP]. 0011 DRUN 1011 LPU_RUN 1100 LPU_STOP 1110 LPU_STANDBY
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 KEY	This field provides the key to change the target mode. Any write access to the register with a value different from the keys is always ignored. Read access will always return the inverted key KEY: 0101_1010_1111_0000 (0x5AF0) INVERTED KEY: 1010_0101_0000_1111 (0xA50F)

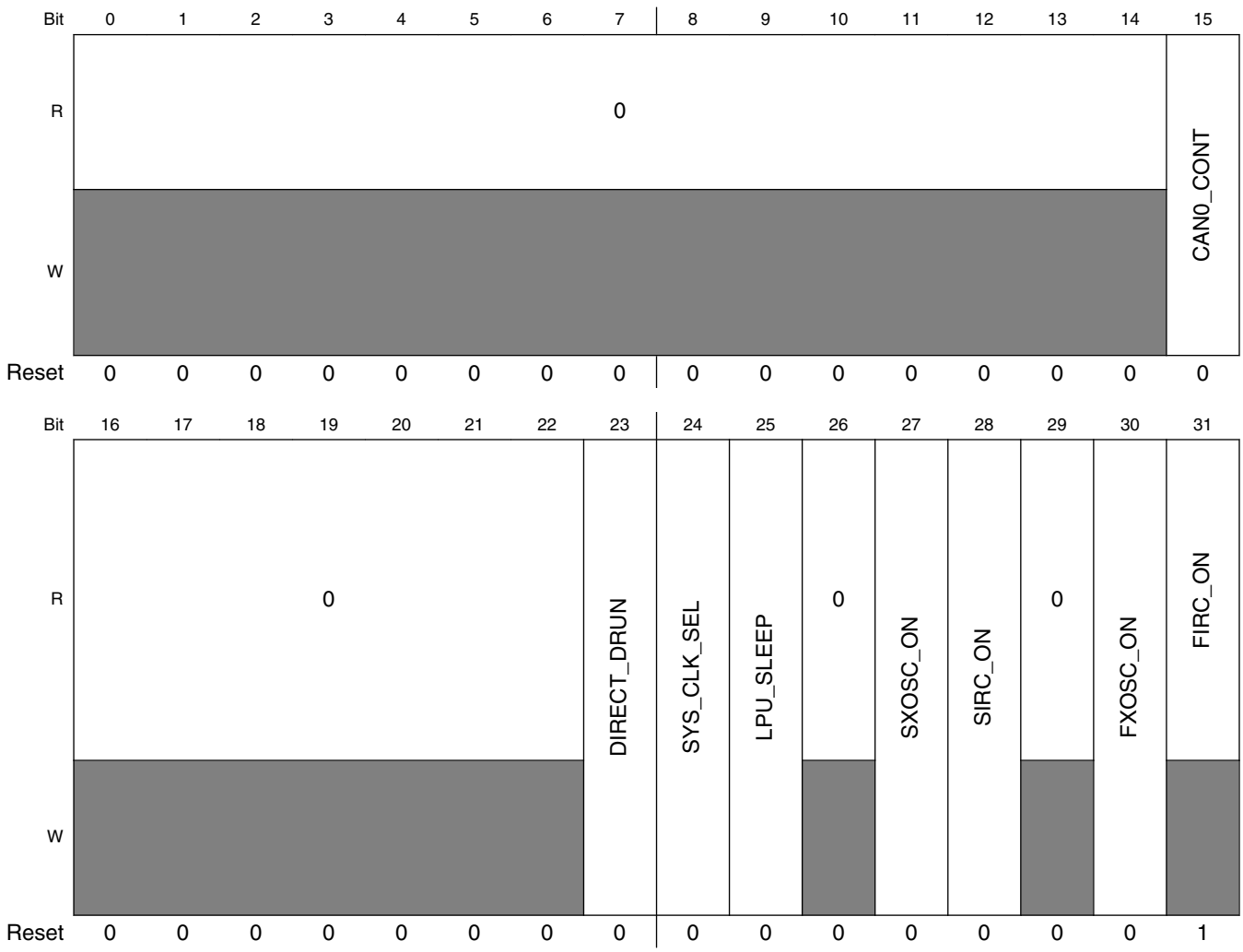
8.11.3 LPU Run Configuration Register (LPU_RUN_CF)

This register configures system behavior during LPU_RUN.

NOTE

LPU_RUN_CF[SYS_CLK_SEL] should never be changed from 1 to 0 in DRUN/RUNx modes. Also, if LPU_RUN_CF[SYS_CLK_SEL] is selecting FXOSC then LPU_RUN_CF[FXOSC_ON] should also be SET.

Address: 0h base + 4h offset = 4h



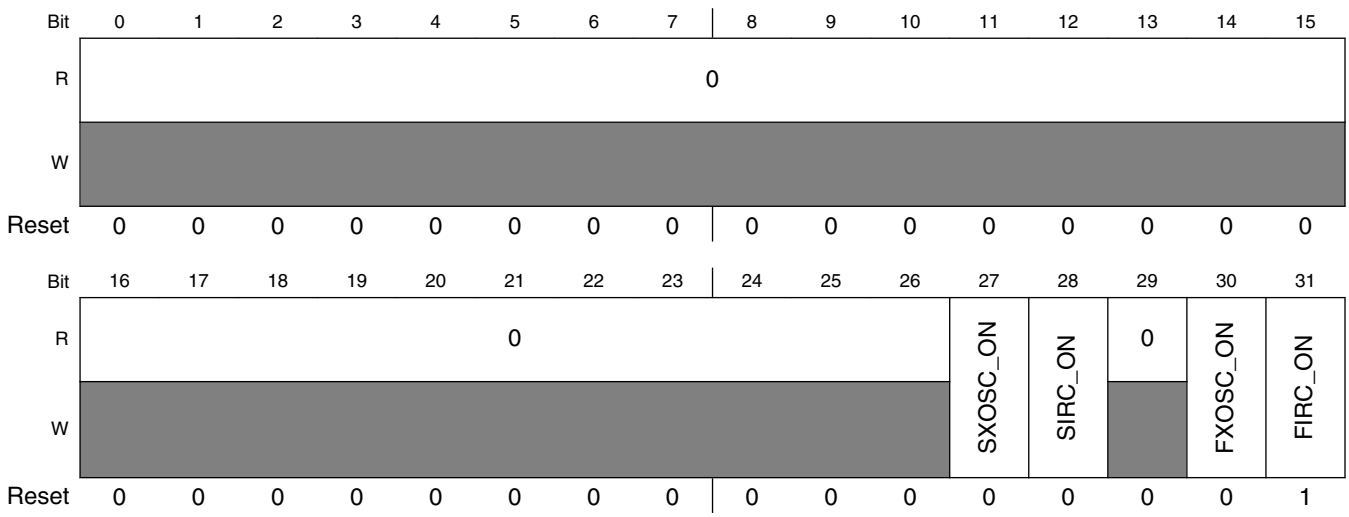
LPU_RUN_CF field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CAN0_CONT	CAN0 continuous operation across LPU modes. CAN CHI frequency will switch from FS80 to LPU_SYS_CLK which will require reconfiguration thus inhibiting continuous operation across LPU modes. 0 Continuous CAN0 operation is not supported 1 Continuous CAN0 operation is supported. CAN0 needs to be operated by selected FXOSC as the source of its CHI clock by configuring MC_CGM_ACx
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 DIRECT_DRUN	Transition to DRUN on LPU_STANDBY exit 0 Switch to LPU_RUN on exit from LPU_STANDBY 1 Switch to DRUN on exit from LPU_STANDBY
24 SYS_CLK_SEL	System clock selection in LPU RUN mode 0 Select FIRC clock source 1 Select FXOSC clock source
25 LPU_SLEEP	Core sleep request during LPU RUN 0 No core sleep request 1 Core sleep request
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 SXOSC_ON	SXOSC control during LPU_RUN 0 SXOSC clock source off 1 SXOSC clock source on
28 SIRC_ON	SIRC control during LPU_RUN 0 SIRC clock source off 1 SIRC clock source on
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 FXOSC_ON	FXOSC control during LPU_RUN 0 FXOSC clock source off 1 FXOSC clock source on
31 FIRC_ON	FIRC oscillator control during LPU_RUN 1 FIRC clock source on 0 Reserved

8.11.4 LPU Stop Configuration Register (LPU_STOP_CF)

This register configures system behavior during LPU_STOP.

Address: 0h base + 8h offset = 8h



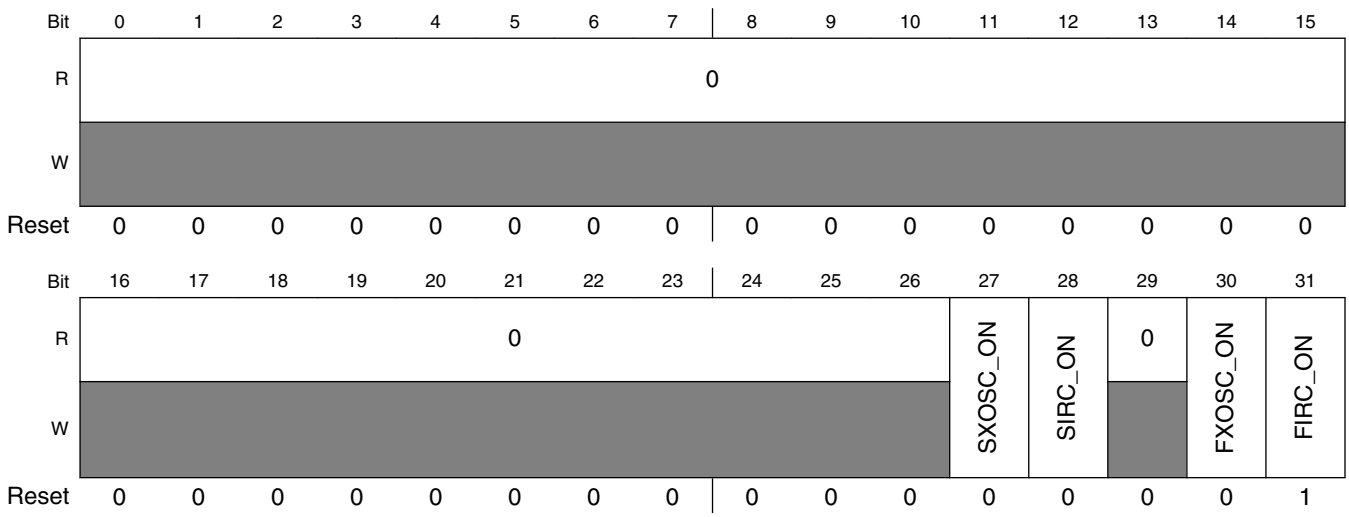
LPU_STOP_CF field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 SXOSC_ON	SXOSC control during LPU_STOP 0 SXOSC clock source off 1 SXOSC clock source on
28 SIRC_ON	SIRC control during LPU_STOP 0 SIRC clock source off 1 SIRC clock source on
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 FXOSC_ON	FXOSC control during LPU_STOP 0 FXOSC clock source off 1 FXOSC clock source on
31 FIRC_ON	FIRC oscillator control during LPU_STOP 0 FIRC clock source off 1 FIRC clock source on

8.11.5 LPU Standby Configuration Register (LPU_STANDBY_CF)

This register configures system behavior during LPU_STANDBY.

Address: 0h base + Ch offset = Ch



LPU_STANDBY_CF field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 SXOSC_ON	SXOSC control during LPU_STANDBY 0 SXOSC clock source off 1 SXOSC clock source on
28 SIRC_ON	SIRC control during LPU_STANDBY 0 SIRC clock source off 1 SIRC clock source on
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 FXOSC_ON	FXOSC clock source control during LPU_STANDBY 0 FXOSC clock source off 1 FXOSC clock source on
31 FIRC_ON	FIRC oscillator control during LPU_STANDBY 0 FIRC clock source off 1 FIRC clock source on

8.11.6 LPU Status Register (LPU_SR)

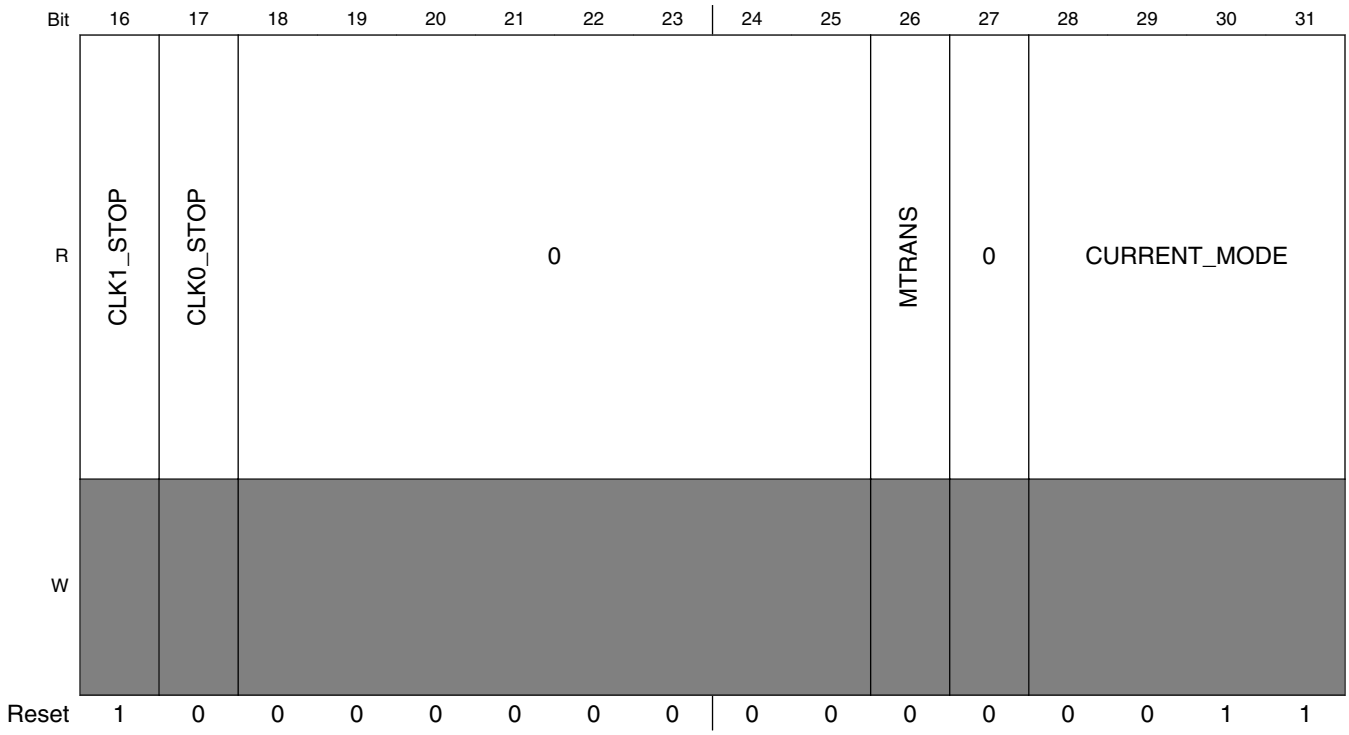
This register provides the system status during LPU mode.

NOTE

The current mode would be updated as the target mode during the mode transition is ongoing i.e., LPU_SR[CURRENT_MODE] would be updated during LPU_SR[MTRANS] = 1.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0										PD2_PWR_STAT	SXOSC_STAT	SIRC_STAT	0	FXOSC_STAT	FIRC_STAT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1



LPU_SR field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 PD2_PWR_STAT	Ensures the main domain (power domain 2) is power gated completely. 0 Main domain is power gated. 1 Main domain is powered
11 SXOSC_STAT	SXOSC clock source status 0 SXOSC clock source off 1 SXOSC clock source on
12 SIRC_STAT	SIRC clock status 0 SIRC clock source off 1 SIRC clock source on
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 FXOSC_STAT	FXOSC clock source status 0 FXOSC clock source off 1 FXOSC clock source on
15 FIRC_STAT	FIRC clock status 0 RC Oscillator off 1 RC Oscillator on

Table continues on the next page...

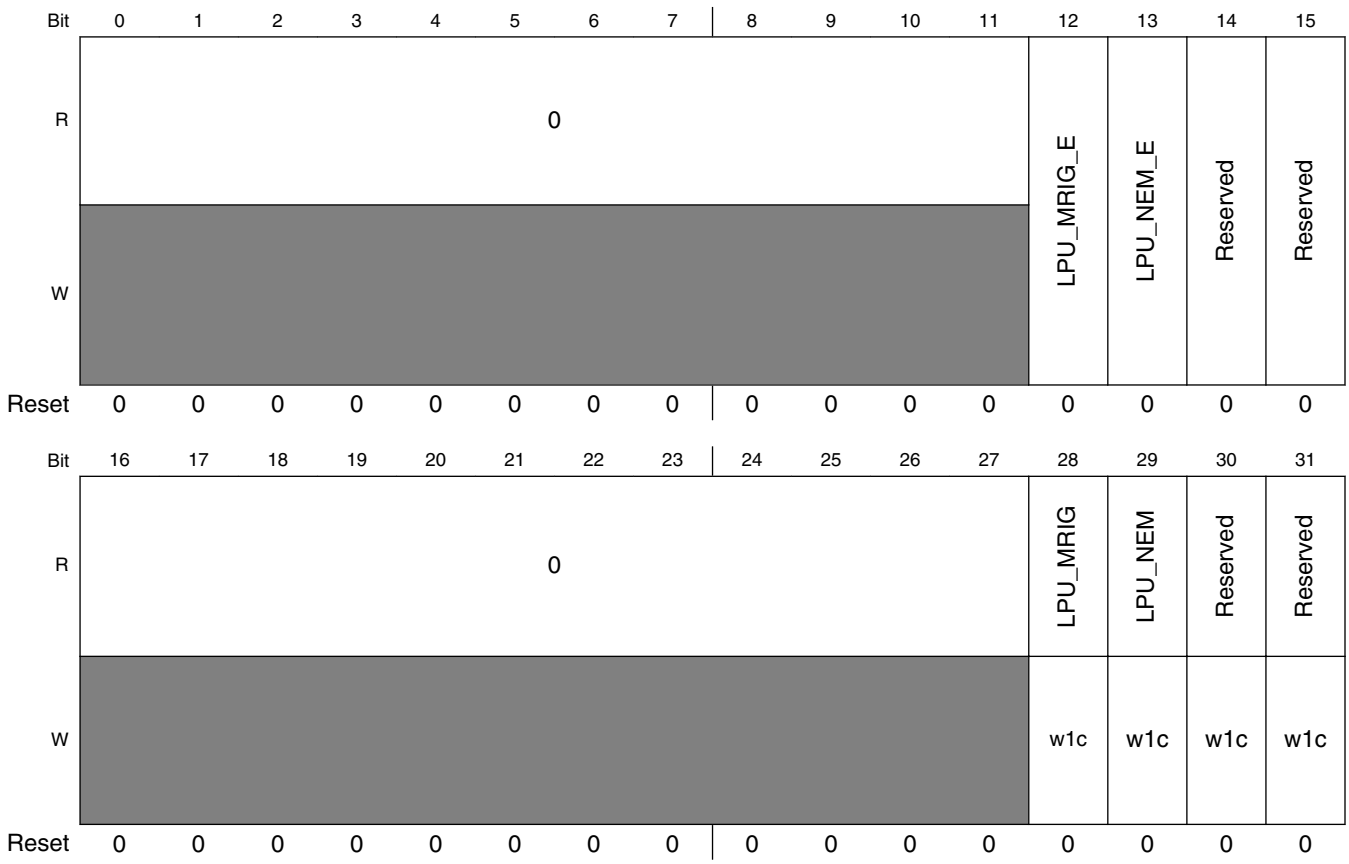
LPU_SR field descriptions (continued)

Field	Description
16 CLK1_STOP	FXOSC clock status during LPU mode 0 FXOSC clock selected as system clock 1 FXOSC clock not selected as system clock
17 CLK0_STOP	FIRC status during LPU mode 0 FIRC clock selected as system clock 1 FIRC clock not selected as system clock
18–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 MTRANS	LPU mode transition 0 No LPU mode transition 1 LPU mode transition is in progress
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 CURRENT_ MODE	Current mode of the chip 0000 RESET 0001 Reserved 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 Reserved 1001 Reserved 1010 STOP0 1011 LPU_RUN 1100 LPU_STOP 1101 STANDBY0 1110 LPU_STANDBY 1111 Reserved

8.11.7 LPU Interrupt control register (LPU_ICR)

This register controls whether an event generates an interrupt. It also provides the interrupt status.

Address: 0h base + 14h offset = 14h



LPU_ICR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 LPU_MRIG_E	LPU mode request ignored interrupt enable 0 Disable interrupt 1 Enable interrupt
13 LPU_NEM_E	LPU non-exist mode interrupt enable 0 Disable interrupt 1 Enable interrupt

Table continues on the next page...

LPU_ICR field descriptions (continued)

Field	Description
14 Reserved	This field is reserved.
15 Reserved	This field is reserved.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 LPU_MRIG	LPU mode request ignored This field is set if during LPU_RUN the core tries to enter a low-power mode, that is, LPU_STOP or LPU_STANDBY, while the interrupt is pending. The core is still able to go to DRUN mode, however. 0 Mode request ignored interrupt is not pending 1 Mode request ignored interrupt is pending
29 LPU_NEM	LPU non-exist mode interrupt This field is set if the mode transition request through the LPU_MCTL register is to a nonexistent mode, that is, not one of the four valid modes: DRUN, LPU_RUN, LPU_STOP, or LPU_STANDBY. The field is cleared by writing 1 to it. 0 Non-exist mode interrupt is not pending 1 Non-exist mode interrupt is pending
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

8.11.8 Module Disable (LPU_MDIS)

If a module is disabled in RUN mode using ME_PCTL, it is suggested to configure MDIS to 1 in LPU modes as by default the clock to that module will be enabled while transitioning to LPU mode.

NOTE

Module system bus clocks are gated in LPU_STOP mode without any option of keeping them ON.

NOTE

During transition from RUNx mode to LPU_RUN mode, peripherals clock gated with LPU_MDIS will observe few clock cycles. Hence it is recommended to check and clear any pending interrupts in LPU_RUN mode. For example, it is possible to have a MEMU interrupt flag set due to some free running clock cycles during RUNx to LPU transition.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						eMIOS_0	CMP_2	CMP_1	CMP_0	MEMU_1	BCTU	CAN0	LIN_0	SPI_0	ADC_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LPU_MDIS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 eMIOS_0	eMIOS_0 is disabled in RUN mode using ME_PCTL, it is suggested to configure MDIS to 1 in LPU modes as by default the clock to that eMIOS_0 will be enabled while transitioning to LPU mode. 0 Module clock is enabled in LPU_RUN or LPU_SLEEP mode. 1 Module clock is disabled in LPU_RUN or LPU_SLEEP mode.

Table continues on the next page...

LPU_MDIS field descriptions (continued)

Field	Description
23 CMP_2	CMP_2 is disabled in RUN mode using ME_PCTL, it is suggested to configure MDIS to 1 in LPU modes as by default the clock to that CMP_2 will be enabled while transitioning to LPU mode. 0 Module clock is enabled in LPU_RUN or LPU_SLEEP mode. 1 Module clock is disabled in LPU_RUN or LPU_SLEEP mode.
24 CMP_1	CMP_1 is disabled in RUN mode using ME_PCTL, it is suggested to configure MDIS to 1 in LPU modes as by default the clock to that CMP_1 will be enabled while transitioning to LPU mode. 0 Module clock is enabled in LPU_RUN or LPU_SLEEP mode. 1 Module clock is disabled in LPU_RUN or LPU_SLEEP mode.
25 CMP_0	CMP_0 is disabled in RUN mode using ME_PCTL, it is suggested to configure MDIS to 1 in LPU modes as by default the clock to that CMP_0 will be enabled while transitioning to LPU mode. 0 Module clock is enabled in LPU_RUN or LPU_SLEEP mode. 1 Module clock is disabled in LPU_RUN or LPU_SLEEP mode.
26 MEMU_1	MEMU_1 is disabled in RUN mode using ME_PCTL, it is suggested to configure MDIS to 1 in LPU modes as by default the clock to that MEMU_1 will be enabled while transitioning to LPU mode. 0 Module clock is enabled in LPU_RUN or LPU_SLEEP mode. 1 Module clock is disabled in LPU_RUN or LPU_SLEEP mode.
27 BCTU	BCTU is disabled in RUN mode using ME_PCTL, it is suggested to configure MDIS to 1 in LPU modes as by default the clock to that BCTU will be enabled while transitioning to LPU mode. 0 Module clock is enabled in LPU_RUN or LPU_SLEEP mode. 1 Module clock is disabled in LPU_RUN or LPU_SLEEP mode.
28 CAN0	CAN0 is disabled in RUN mode using ME_PCTL, it is suggested to configure MDIS to 1 in LPU modes as by default the clock to that CAN0 will be enabled while transitioning to LPU mode. 0 Module clock is enabled in LPU_RUN or LPU_SLEEP mode. 1 Module clock is disabled in LPU_RUN or LPU_SLEEP mode.
29 LIN_0	LIN_0 is disabled in RUN mode using ME_PCTL, it is suggested to configure MDIS to 1 in LPU modes as by default the clock to that LIN_0 will be enabled while transitioning to LPU mode. 0 Module clock is enabled in LPU_RUN or LPU_SLEEP mode. 1 Module clock is disabled in LPU_RUN or LPU_SLEEP mode.
30 SPI_0	SPI_0 is disabled in RUN mode using ME_PCTL, it is suggested to configure MDIS to 1 in LPU modes as by default the clock to that SPI_0 will be enabled while transitioning to LPU mode. 0 Module clock is enabled in LPU_RUN or LPU_SLEEP mode. 1 Module clock is disabled in LPU_RUN or LPU_SLEEP mode.
31 ADC_0	ADC_0 is disabled in RUN mode using ME_PCTL, it is suggested to configure MDIS to 1 in LPU modes as by default the clock to that ADC_0 will be enabled while transitioning to LPU mode. 0 Module clock is enabled in LPU_RUN or LPU_SLEEP mode. 1 Module clock is disabled in LPU_RUN or LPU_SLEEP mode.

Chapter 9
Clocking Overview

9.1 Introduction

This chapter describes the architecture for the system level clocks and includes the following information:

- Dynamic frequency scaling
- System clock specifics
- Clock architecture
- Clock sources
- Clock monitoring
- Programmable clock dividers
- Clock control registers
- Progressive clock switching

NOTE

This chapter covers only clocks that are generated on the device. Protocol clocks that are provided by external devices are covered in the respective chapters where they are used.

The system clock can be configured as frequency modulated clock to reduce electromagnetic emissions, while having a precise clock for timers and communication functions[1]. There is also the ability of the chip to enter a low power mode and utilize the low power clocks.

The device boots from the internal 16 MHz RC oscillator (FIRC), and uses this as a backup clock in the event of a PLL or oscillator failure. The backup clock is enabled in the Mode Entry Module (MC_ME) (see the "Mode Entry Module (MC_ME) Overview chapter for details).

System clocks can be generated from the three sources:

1. Using the frequency modulated clock is not recommended for LINFlexD.

- Fast external crystal oscillator 8 – 40 MHz (FXOSC)
- Fast internal RC oscillator 16 MHz (FIRC)
- Frequency-modulated phase-locked loop (FMPLL)

Additionally, there are two low power oscillators:

- Slow internal RC oscillator 128 KHz (SIRC)
- Slow external crystal oscillator 32 KHz (SXOSC)

Internal system clocks are generated from either PLL_PHI_0, FIRC, or FXOSC. There is one clock selector which is used exclusively for system clock source selection. The system clocks along with PLL_PHI_0, FIRC, FXOSC, SIRC, SXOSC, can be selected to drive system peripherals depending on the configuration of the Auxillary Clock Selectors in the Clock Generation Module (MC_CGM), and selectors based locally in the modules. There are six Auxillary Clock Selectors which allow developers to drive peripherals with a specific clock source (see Figure 9-1).

The MPC5746C has a Clock Monitor Unit (CMU) that is used to check the integrity of the clocks making sure that their frequencies stay within necessary operating limits. If the CMU detects an issue with the monitored clocks, an interrupt or system reset could be generated (depending on FCCU configuration).

NOTE

The auxilary clock multiplexers (for example, MC_CGM_AC*n*_SC[SELCTL]) should be programmed to select the clock source before the corresponding module is enabled.

9.2 System clock generation

The MPC5746C supports seven synchronous clock nodes to support Dynamic Frequency Scaling (DFS). These groups can be classified into 3 categories:

- Fixed Frequency Group (F160, F80, F40, F20): This group will not scale down during DFS operation. It implies all the modules will continue receiving the same frequency (for example, PIT timers are operating at F40MHz and would not scale down).
- Scalable Frequency Group (S160, S80, S40): This group will scale down during DFS operation (for example, Z4 core is operating at S160 and would scale down as per configuration in MC_CGM_SC_DC*n* registers.
- Flexible Group (FS80): This group should be configured to scale down along with the Scalable group during Linear DFS and should be kept at a fixed frequency during Non-Linear DFS.

Each module can have clocks coming from different groups. See Table 9-6 for peripheral clocking.

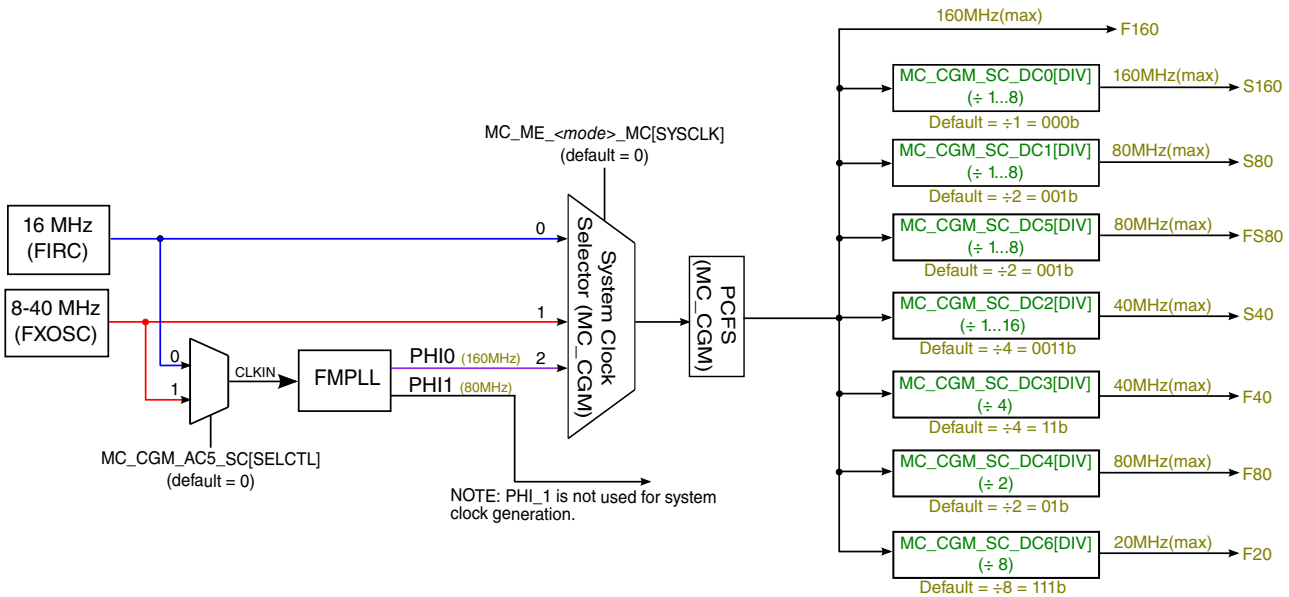


Figure 9-1. System clock generation

9.3 Dynamic Frequency Scaling

9.3.1 Overview

Dynamic Frequency Scaling (DFS) enables the core and system frequency to be reduced during RUN mode, while keeping frequency of key peripherals constant. This enables the power consumption of the device to be temporarily reduced when processing needs are low.

There is a considerable amount of power savings as frequency is reduced for all three cores, AXBS, flash memory, SRAM, PBRIDGE and bus interface unit of modules. During this mode, the frequency for timers like PIT, SWT, eMIOS, and baud rates for modules like LIN, DSPI, are kept constant.

The clocks scaled during DFS are the following:

- S40
- S80
- S160
- FS80

NOTE

All frequencies in this chapter are based on assumption that PLL PHI0 is 160MHz. If PLL PHI0 is at a different clock frequency the system clock nodes will vary based on divider configurations in MC_CGM_SC_DCn

There are two DFS modes supported:

- Linear DFS (LDFS)
  - This mode is entered by writing MC _CGM_SC_DIV_RC[SYS_DIV_RATIO_CHNG] = 0. In LDFS, the system clock divider ratios will not change when the next system clock divider configuration is updated.
- Non_Linear DFS (NLDFS)
  - This mode is entered by writing MC _CGM_SC_DIV_RC[SYS_DIV_RATIO_CHNG] = 1. In NLDFS, the system clock divider ratios will change when the next system clock divider configuration is updated.

The below table gives frequency scaling during all DFS modes.

NOTE

FS80 must not be configured to any frequency less than that of F40 group.

Table 9-1. Dynamic Frequency scaling

Clock nodes	RUN - Full Power	Linear DFS	NLDFS80	NLDFS40	NLDFS20
F160	160	160	160	160	160
S160	160	80	80	40	20
S80	80	40	80	40	20
S40	40	20	20	20	10
F40	40	40	40	40	40
F80	80	80	80	80	80
FS80	80	40	80	80	80
F20	20	20	20	20	20

All the modules which are clocked from a scalable frequency group will scale down in both the modes (see System clock generation).

9.3.2 Linear DFS Configuration

In this mode, frequency is scaled down while keeping the ratio between master (for example, FlexRay, DMA) and AXBS as same. For example, in full run mode Z2 is at 80 MHz and AXBS is at 160 MHz, thus ratio is 1:2. In Linear DFS mode Z2 can be scaled down to 40 MHz and AXBS to 80 MHz, thus maintaining a 1:2 ratio. Transitioning between frequencies occurs seamlessly, such that transactions can continue without halting AXBS.

Following is the frequency configurations supported in Linear DFS (all frequencies are in MHz).

Table 9-2. Linear DFS supported frequencies

	System (Z4 core, AXBS, SRAM, flash memory)	Z2	eDMA	HSM	ENET	FlexRay	PBRIDGE
Case 1: Full RUN	160	80	80	80	80	80	80
Case 2: Linear DFS	80	40	40	40	—[1]	—[2]	40

1. Linear DFS feature not supported.
2. The Flexray can be used in the Linear DFS mode only when the FXOSC is selected as the Flexray PE clock source.

Suggested configuration sequence for Linear DFS

1. Configure MC_CGM_SC_DC0[DIV] = 01b, to scale down S160 group to generate 80 MHz
2. Configure MC_CGM_SC_DC1[DIV] = 0011b, to scale down S80 group to generate 40 MHz
3. Configure MC_CGM_SC_DC2[DIV] = 0111b, to scale down S40 group to generate 20 MHz
4. Configure MC_CGM_SC_DC5[DIV] = 011b, to scale down FS80 group to generate 40 MHz
5. Configure MC_CGM_SC_DIV_RC[SYS_DIV_RATIO_CHNG] = 0 to disable HALT/ACK handshake with AXBS.
6. Configure MC_ME_PCTL*n* to disable ENET (MC_ME_PCTL15) to save dynamic power.
7. Configure MC_CGM_DIV_UPD_TRIG any non-zero value.

9.3.3 Non-Linear DFS Configuration

In this mode frequency is scaled down and also the ratio between master and AXBS can be varied. For example, in full run mode ENET is at 80 MHz and AXBS is at 160 MHz, thus ratio is 1:2. In Non-Linear DFS mode ENET can be maintained at 80 MHz and AXBS scaled to 80 MHz, thus changing ratio to 1:1. It is required to put the AXBS into a quiescent state such that on a DFS request all ongoing transactions at the AXBS slave are completed and an acknowledgement is issued to the MC_CGM to change frequencies.

Following is the frequency configurations supported in Non-Linear DFS.

Table 9-3. Non-Linear DFS supported frequencies

	System (Z4 core, AXBS, SRAM, flash memory)	Z2	eDMA	HSM	ENET	FlexRay	PBRIDGE
Case 1:Full RUN	160	80	80	80	80	80	80
Case 3: NLDFS80	80	80	80	40	80	80	80
Case 4: NLDFS40	40	40	80	20	80	80	80
Case 5: NLDFS20	20	20	80	10	80	80	80

Suggested configuration sequence for Non-Linear DFS

- Configure MC_CGM_SC_DC0[DIV] = 01b, to scale down S160 group to generate 80 MHz
- Configure MC_CGM_SC_DC2[DIV] = 0111b, to scale down S40 group to generate 20 MHz
- Configure AXBS*x*_PRS*n* to set the priorities as per application requirement. It is suggested to provide highest priority for FlexRay.
- Configure MC_CGM_SC_DIV_RC[SYS_DIV_RATIO_CHNG] = 1 to enable HALT/ACK handshake with AXBS. Configure MC_CGM_DIV_UPD_TRIG any non-zero value.

If lowest frequency Non-Linear DFS is chosen, the restrictions are same as that of lowest frequency DFS case.

NOTE

The modules working on FS80 group will not be scaled down in all NLDFS modes.

9.3.4 DFS Limitations

There are some frequency requirements mandated by the protocol which should be considered while selecting DFS mode for an application.

Refer Peripheral Clock Restrictions for more details

Table 9-4. DFS Limitations

Mode	Limitation
NLDFS80	No restriction. All modules can operate well.
LDFS	Refer Peripheral Clock Restrictions

9.3.5 NLDFS Abort on Interrupt

When the software initiates NLDFS, the crossbar including the masters are halted before the system frequency is dynamically switched. It may take a while for all masters to halt together. However, the NLDFS can be aborted in between using following critical interrupts.

- NMI to Z4 and Z2
- Critical interrupts to Z4 and Z2
- Machine check request to Z4 and Z2
- SWT0 Interrupt
- SWT1 Interrupt
- HSM_SWT interrupt
- Flexray FIFO interrupts such as, *FR_GIFER[FAFAIF]* or *FR_GIFER[FAFBIF]*
- MAC0 underrun interrupt, *MAC 0 Transmit FIFO Underrun*

The enabling and disabling of HALT abort feature can be controlled by MC_CGM_SC_DIV_RC[HALT_ABORT_DIS], System Clock Divider Ratio Change Register (MC_CGM_SC_DIV_RC)

9.4 Peripheral clock generation

9.4.1 Overview

Each module requires the following clocks to operate (see Figure 9-2):

- HCLK – This is the AHB interface clock of modules which are master or slaves on AXBS. This clock is gated in low power modes, or if the module is disabled and is supported by MC_ME_PCTL configuration.
- Peripheral CLK – This clock is required by modules for internal state machines and processing of data.
- BIU CLK – Bus interface unit clock. This clock is required for register read write configurations. It is available only when a module is accessed for configurations and else it is disabled. Some modules operate its bus interface unit on peripheral clock and corresponding column is marked as '—' for these modules (see System Clock Distribution for details).
- Memory CLK - Memory of the modules operate on this clock.
- Baud_CLK/Auxiliary CLK - This clock is provided to generate external interface clock or if there is any special clock requirement

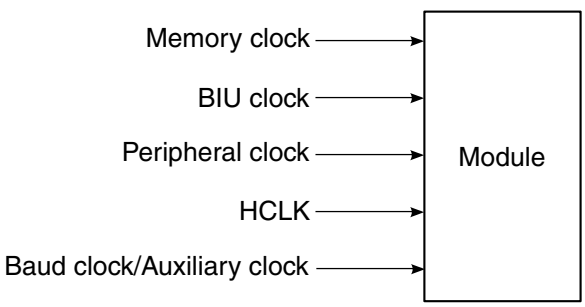


Figure 9-2. General module clocking

9.4.2 Enabling/Disabling of peripherals and clock sources

Disabling of peripherals and clock sources: Software should take care that secondary (auxiliary) clock sources selected for a peripheral are not disabled during any mode transition preceding a mode transition to disable the peripheral. For Example: In FlexCAN when FXOSC is selected for the CAN Engine Clock Source (CAN_CTRL1[CLKSRC]), before disabling FlexCAN Peripheral Clock the user must ensure FXOSC is enabled in the current mode prior to the target mode transition that will disable the FlexCAN.

Enabling of peripherals and clock sources: Software should take care that secondary (auxiliary) clock sources are enabled for the target mode before initiating the target mode transition to enable the peripheral.

9.4.3 Communication interfaces

9.4.3.1 FlexCAN clocking

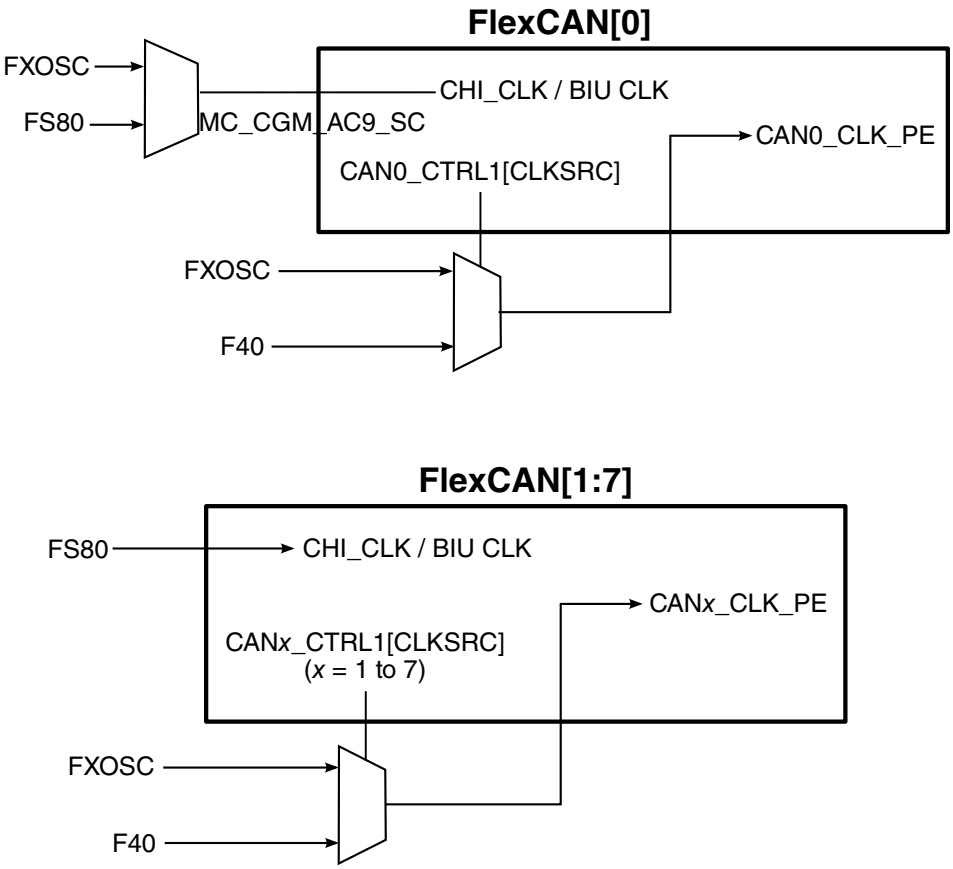
The FlexCAN Protocol Engine clock (PE clock) has an option to be sourced from either the F40 or the external oscillator (FXOSC). Clock selection is controlled by CAN*x*_CTRL1[CLKSRC]. Control host interface(CHI) clock of CAN1 to CAN7 is sourced from FS80 group whereas, CAN0 has an option to be sourced from either FXOSC or FS80 group.

NOTE

FlexCAN_1-7 CHI clock frequency should not be lower than the PE clock when the system clock is dynamically updated.

FlexCAN0 CHI clock and PE clock can operate in 1:1 frequency ratio. There is an auxiliary mux selector added to ensure seamless operation while entry-exit from LPU subsystem. ReferFlexCAN0 Clocking

Software Guideline: If FXOSC is selected as the FlexCAN_*n* clock source, MC_ME_PCTL[70:77] (corresponding to FlexCAN_0 through FlexCAN_7, respectively) should be configured after MC_ME_GS[S_FXOSC] = 1. For example, if FXOSC is the source for FlexCAN_0, then MC_ME_PCTL70 should be configured after MC_ME_GS[S_FXOSC] = 1. This is required to ensure clock source is available before module is enabled.



NOTE: The multiplexers for all FlexCAN modules are located outside of the module, but are controlled by registers inside the FlexCAN modules.

Figure 9-3. FlexCAN Clock generation

9.4.3.2 FlexRay clocking

The FlexRay protocol clock can be sourced from either FS80 (non-FM PLL) or the external oscillator (FXOSC). Clock selection is controlled by configuring the FlexRay Module Configuration register FR_MCR[CLKSEL] field. Control host interface (CHI) clock of FlexRay is sourced from FS80 group.

- When using the external oscillator clock, a 40 MHz crystal is required for proper operation.
- When using FS80, the MC_CGM_SC_DC5 register must be configured to provide 80 MHz on FS80 for proper operation.
- The frequency of the FS80 source must be double the frequency of the XOSC source (40 MHz) because the PLL has a less stable duty cycle.

- The FlexRay module itself divides the FS80 frequency to provide a steady, 50% duty cycle, 40 MHz clock internally.
- This internal FlexRay divider is driven automatically when FR_MCR[CLKSEL] is 1. The user cannot change this internal divider or select it for the XOSC source.

NOTE

The FlexRay module used in the device has the StopWatch functionality. The OS_tick is the input into the FlexRay module and the source is from the STM and PIT channels.

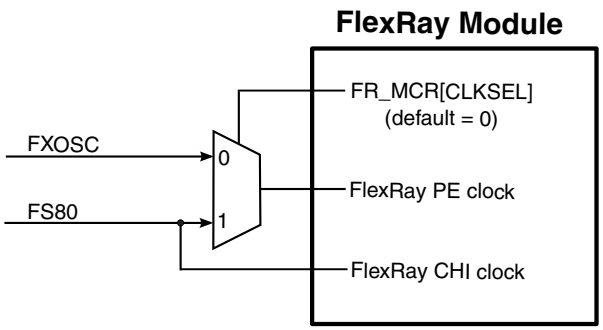
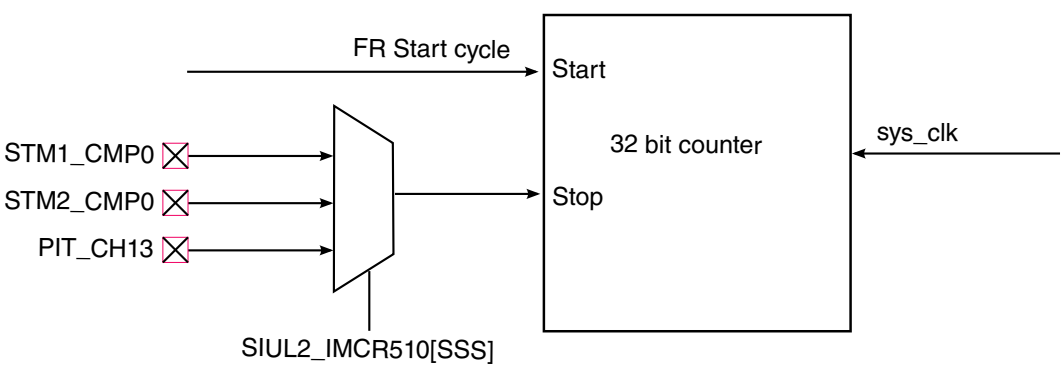


Figure 9-4. FlexRay clock generation

9.4.3.2.1 FlexRay OS tick selection

The following figure shows the OS tick selection configuration which is implemented at the device level outside the FlexRay StopWatch interface (see the StopWatch function section in the "FlexRay" chapter for details).



For detailed description, see I/O Signal Description and Input Multiplexing Tables (excel file) attached to this document.

NOTE

Software Guideline: If FXOSC is selected as the FlexRay clock source, MC_ME_PCTL28 needs to be configured after MC_ME_GS[S_FXOSC] = 1.

9.4.3.3 LINFlexD clocking

LINFlexD requires bus interface unit clock for configurations and a baud clock. LINFlexD belongs to fixed frequency group such that its baud rate remains constant during DFS.

LINFlexD[0] BIU clock is from F20 group. It cannot belong to F80 group as maximum frequency supported in LPU subsytem is 40 MHz. Refer LinFlexD Clocking

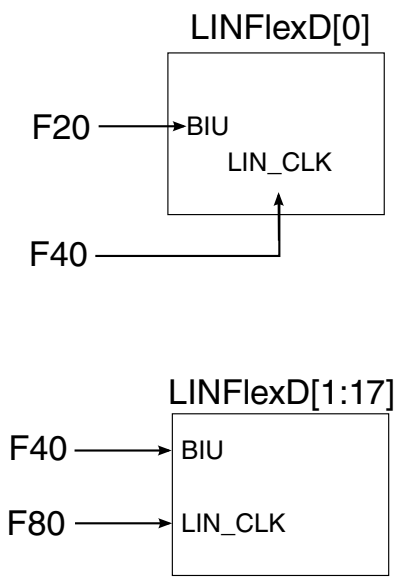


Figure 9-5. LINFLexD Clock generation

9.4.3.4 ENET clocking

ENET HCLK cannot be below 50 MHz to support MII and RMII modes. Either F40 or the Ext_CLK clocks can be selected as the timer clock. The period of this clock should be an integer value (for example, 25 ns if 40 MHz clock frequency is used).

NOTE

Ext_CLK is sourced from the TXD_CLK/RMII_CLK external pad which is TXD_CLK in MII mode.

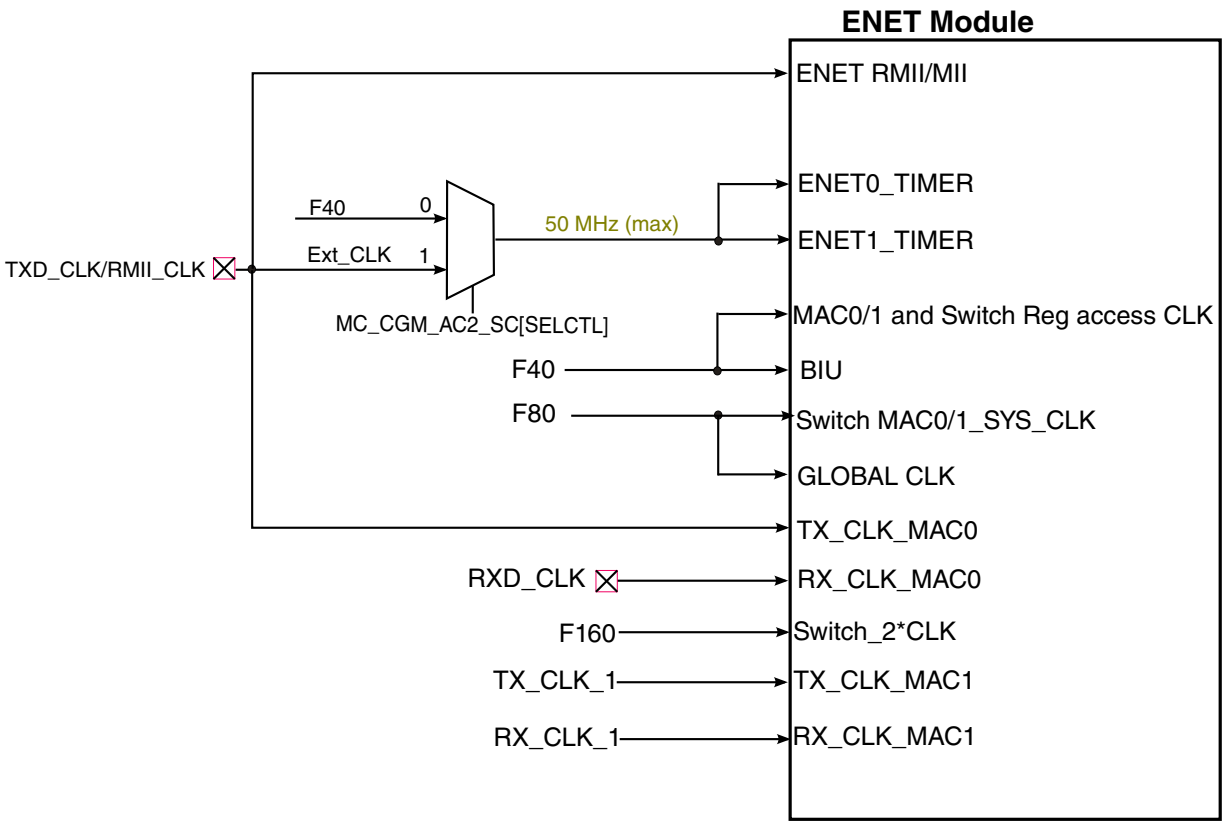


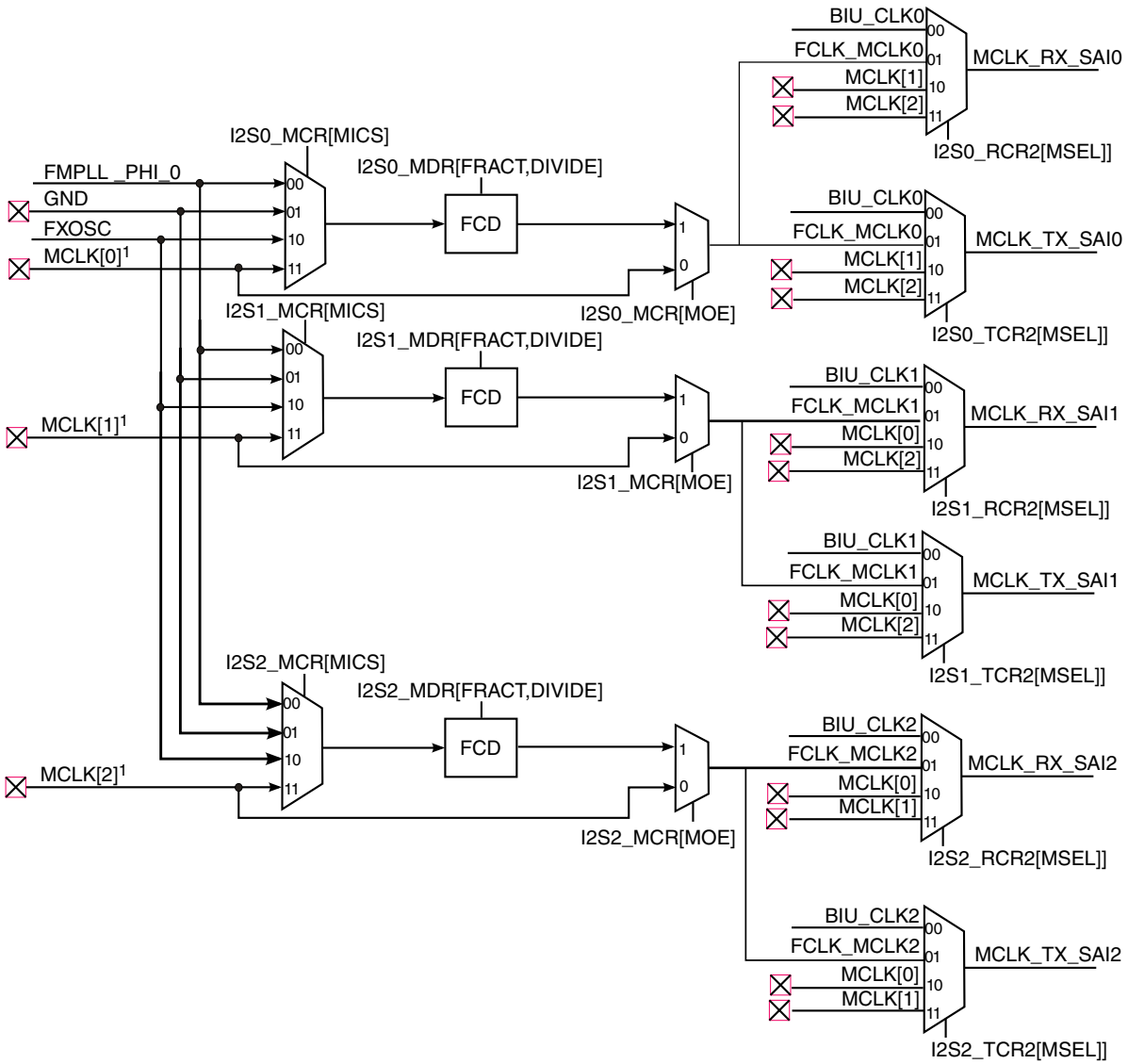
Figure 9-6. ENET Clock generation

9.4.3.5 SAI clocking

Figure 9-7 shows how the SAI master clock (MCLK) is generated via fractional clock divider (FCD) . The input clock to the FCD is limited to 160 MHz. MCLK[2:0] can also come as an input via external pins. Bit clock (BCLK) is half of MCLK and can be generated internally by dividing MCLK or they can be input via external pin. BIU operates at S40.

NOTE

BCLK should never be greater than BIU clock which is S40 as it will lead to underrun or overflow of FIFO inside SAI.



[1]Driven by signals external to the MPC5746C

Figure 9-7. SAI clock generation

9.4.3.6 DSPI/SPI clocking

There are four DSPI (DSPI_[0:3]) and four SPI (SPI_[0:3]) modules on the MPC5746C. DSPI_[0:1] and SPI_[0,3] uses F40 for its baud rate clock, and DSPI_[2:3] and SPI_[1:2] uses F80 for its baud rate clock as given in the following figure.

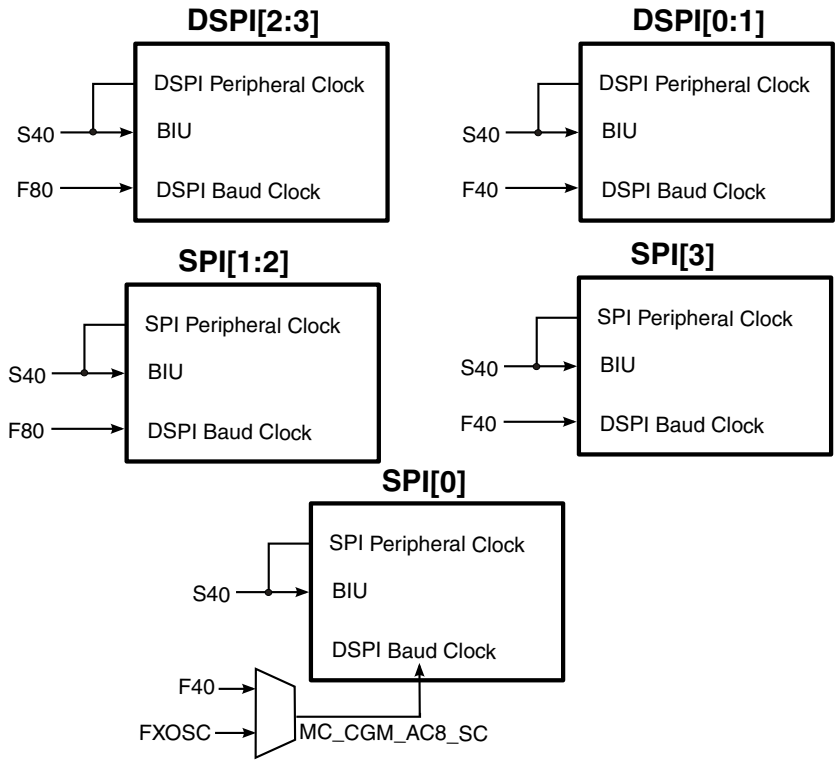


Figure 9-8. DSPI/SPI clock generation

9.4.4 Timers Clock

9.4.4.1 RTC clocking

The RTC_CLK for the RTC-API is generated by selecting one of four clocks as shown in Figure 9-9. If needed, the RTC_CLK can be divided for finer granularity (see RTC Block Diagram).

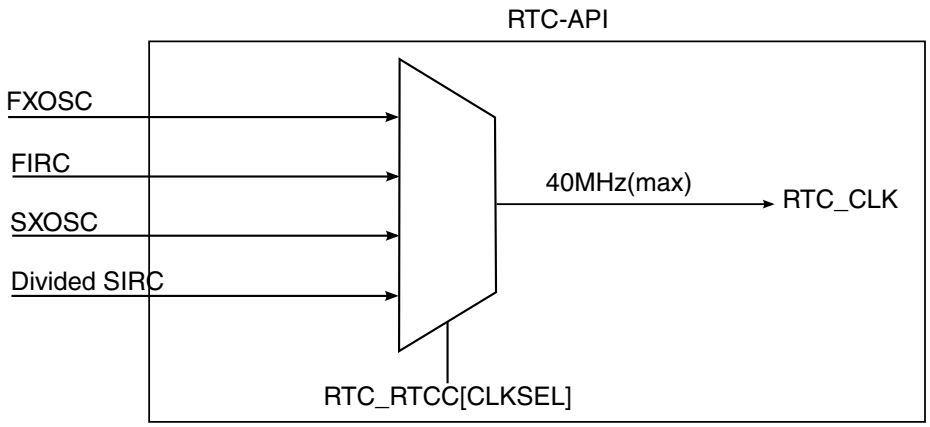


Figure 9-9. RTC Clock generation

NOTE

SIRC_DIV if configured for non zero value, will get reset on any chip reset thus generating 128kHz clock frequency.

9.4.4.2 PIT-RTI clocking

The PIT_RTI contains 16 PIT channels and a single RTI channel. The PIT channels are clocked from the system clock (F40). The clock source for the RTI is the FXOSC.

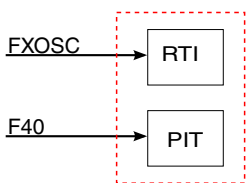


Figure 9-10. PIT Clock generation

NOTE

Software Guideline: If FXOSC is selected as the RTI clock source, MC_ME_PCTL91 needs to be configured after MC_ME_GS[S_FXOSC] = 1. Also, the clock for RTI timers needs to be enabled (PIT_MCR[MDIS_RTI] = 1) after MC_ME_GS[S_FXOSC] = 1.

9.4.4.3 STM clocking

FS80 is selected as the source clock for all STM to allow a constant frequency of 80 MHz while core frequency scales down from 160 MHz to 80 MHz during NLDFS.

NOTE

STM counter should be enabled only after selecting STM_2CLK_CR[CSL].

NOTE

If STM is configured in STOP/STANDBY/LPU modes, FXOSC should be kept ON in target modes (RUNx/DRUN) also, to exit properly

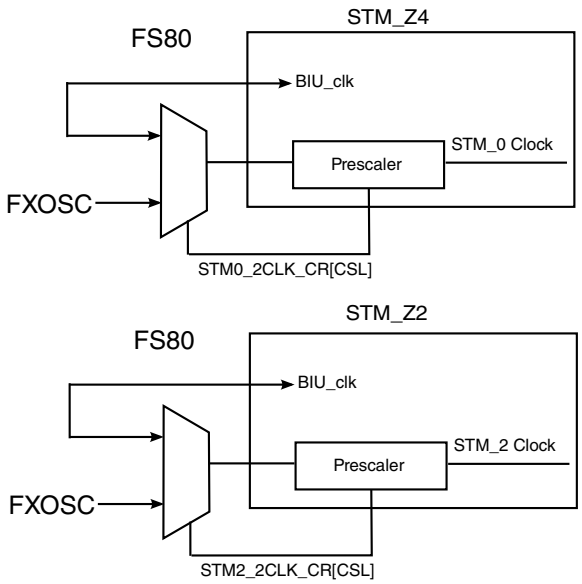


Figure 9-11. STM Clock generation

9.4.4.4 SWT clocking

SIRC is always enabled after reset such that the SWT receives the clock during the reset sequence and avoids a timeout. If a timeout occurs, a destructive reset will be initiated indicating flash memory initialization fails.

NOTE

SWT must be configured only if ME_GS[S_SIRCON] is 1.

NOTE

SIRC should always be kept ON (in DRUN/RUNx modes) if at least one SWT is enabled to allow proper mode transitions.

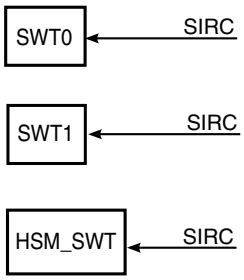


Figure 9-12. SWT clock generation

9.4.5 ADC/BCTU/eMIOS clocking

The ADC clock source is FS80. The ADC can reduce the source clock to generate a 40 MHz clock of the correct tolerance. The ADC, BCTU, and eMIOS are in the same 80 MHz peripheral set to avoid clock domain crossing delays. These modules will not be scaled down across NLDFS modes.

NOTE

The ADC meets the 1 μS conversion time with an 80 MHz clock source. If the clock frequency is higher than 80 MHz, the conversion accuracy is reduced. If a lower frequency is used, conversion time will be longer, but accuracy is not affected. The minimum ADC clock frequency is 1 MHz (see the MPC5746C Data Sheet for timing details).

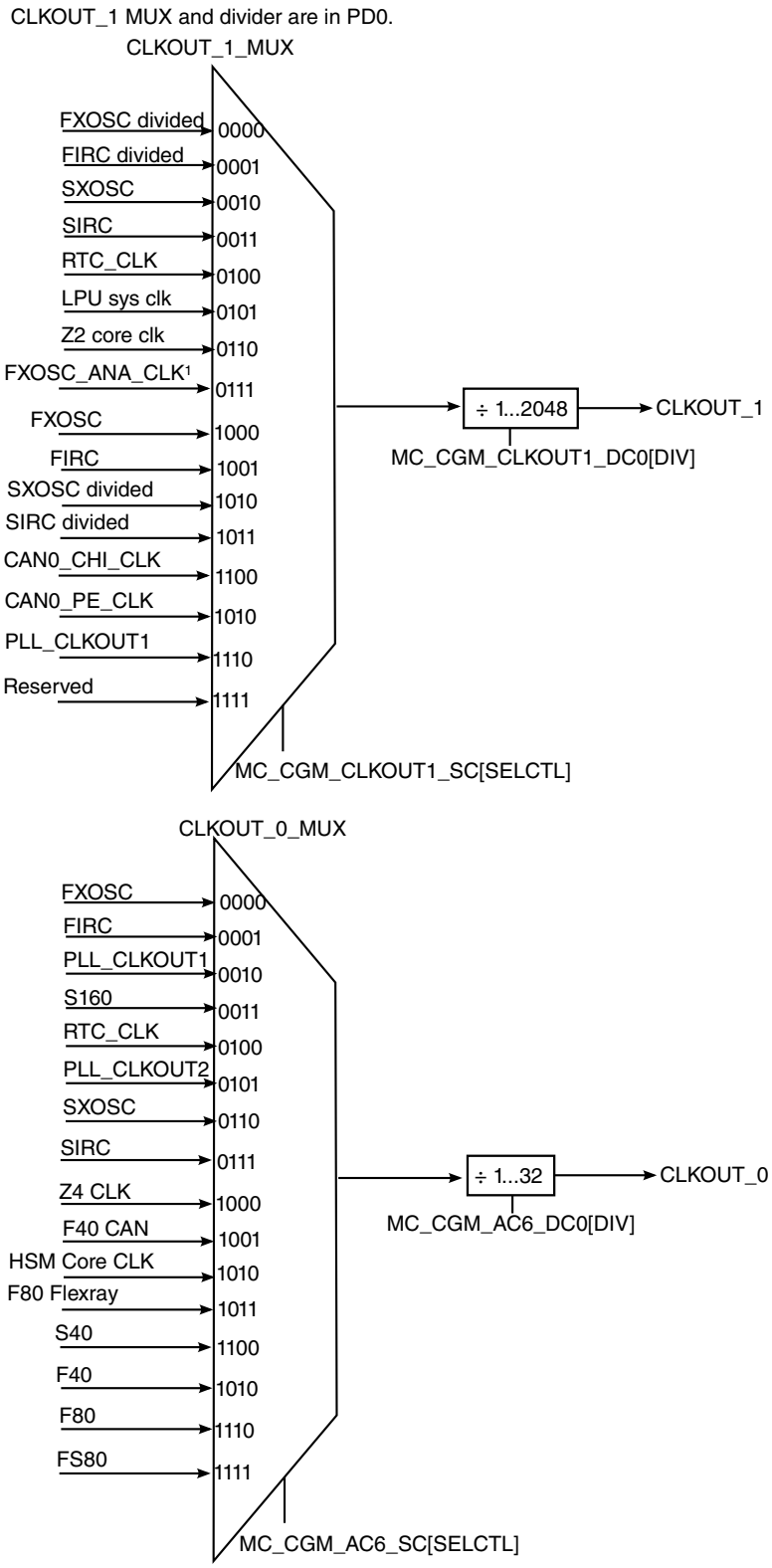
9.4.6 CLKOUT clocking

There are two external clock output pins on the device:

- CLKOUT_0
- CLKOUT_1

CLKOUT_0 clock is available only in RUN, DRUN and STOP modes. However, CLKOUT_0 pin is available in all modes, including STANDBY and LPU modes.

Both CLKOUT_1 clock and CLKOUT_1 pin are available in all modes, including STANDBY and LPU modes.



1. FXOSC_ANA_CLK is an output of analog fast oscillator which is actually fed from XTAL on board. This has same frequency as FXOSC.

Figure 9-13. CLKOUT clock generation

9.4.7 HSM clocking

The HSM Core works at a set frequency of half the system clock (S160). The clock divider for random number generator (TRNG) is located outside the HSM in the CGL, is controlled by the HSM. The HSM contains an internal PIT/RTI block, which uses the FIRC as its clock source. The HSM also contains an internal clock monitor on its input clock.

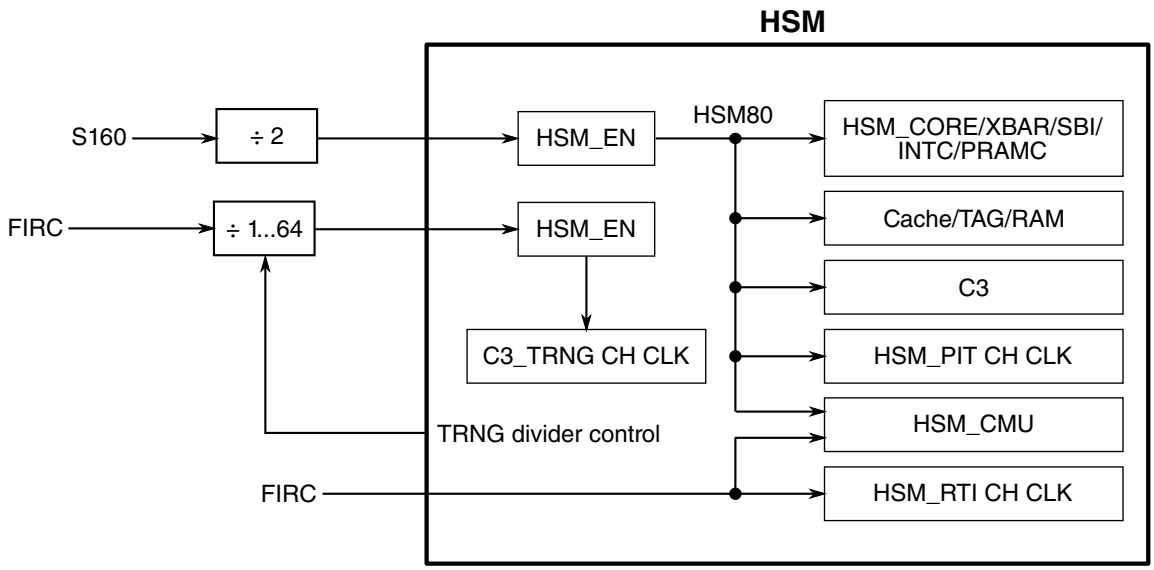


Figure 9-14. HSM clocking

9.4.8 LPU clocking

It is possible to run the Low Power Unit (LPU) from either the FXOSC or FIRC. These frequencies can be reduced using dividers before they are routed to the peripherals in the low power domain (for example, LINFlexD, SPI, ADC, and so on).

NOTE

See the Clocking section in the LPU chapter for clocking details during low power operation.

9.5 MC_CGM registers

The relationship between the output clocks and the MC_CGM registers is shown in Figure 9-1.

NOTE

See Clock Generation Module (MC_CGM) for specifics on register implementation.

9.6 Built-In Self-Test clocking

The device supports user Built-In Self-Test (MBIST) as part of the functional safety concept. The configuration is controlled via the STCU2 module. MBIST can run at start up (before cores exit reset), or initiated by application during run time. It is possible to run Built-In Self-Test from the FIRC or PLL_PHI1 clock (limited to 40 MHz). At start up, the selection of the BIST clock is configured in the DCF record, STCU_PLL_CFG (default is the FIRC). As cores are not running during MBIST, the PLL setup is controlled with hardware. During Run time, the BIST clock selection is done by the application by writing in STCU2_RUNSW Register.

NOTE

The STCU2 registers read/write is done on S40 clock.

9.7 System Clock Distribution

The maximum frequency of operation for the device system level clocks is shown in Table 9-5 and Table 9-6.

The AXBS are clocked from the S160 group. There is no mode where any of the masters or slaves can work at a frequency higher than the AXBS frequency.

The PBRIDGE clock cannot be scaled below 40 MHz. It is clocked from FS80 group, there is no mode where any of the slaves can work at a frequency higher than the PBRIDGE frequency. As there are modules which have their BIU clocked from F40 group (except for PIT and LINFlexD), this limits PBRIDGE to 40 MHz.

Dividers corresponding to fixed frequency groups (for example, F20, F40, F80, F160) should not be modified in any DFS mode.

NOTE

The user is responsible to verify that these values are not exceeded.

In Table 9-5 and Table 9-6 the Peripheral clock will also serve the purpose of BIU clock for the modules in which BIU clock is not defined

Table 9-5. Maximum clock frequencies - AXBS master/slave

Module	HCLK	Peripheral CLK	BIU	Memory CLK	BAUD_CLK/Auxiliary CLK
Z4	S160	—	—	—	—
Z2	S80	—	—	—	—
DMA	FS80	—	—	FS80	—
HSM	HSM80	HSM80	HSM80	—	See HSM clock diagram for details.
FlexRay	FS80				
ENET	F80	F80	F40	F80	Input clock from PAD
PBRIDGE_A, PBRIDGE_B	FS80	FS80	—	—	—
PFLASH_CTRL	S160	—	S80	S160	—
PRAM_CTRL	S160	S80	S80	S160	—

NOTE

In this document, few module refer to ipg_clk as the system clock. Please refer the following table to see the clock mapping for each module.

Table 9-6. Maximum clock frequencies - Peripherals

Module	HCLK	Peripheral CLK	BIU	Memory CLK	BAUD_CLK/Auxiliary CLK
SSCM	—	S40	S40	—	—
Flash	—	S160	S40	—	—
MC_ME	—	S40	S40	—	—
MC_RGM	—	FIRC (16 MHz)	S40	—	—
MC_PCU	—	FIRC (16 MHz)	S40	—	—
MC_CGM	—	—	S40	—	—
FIRC	—	S40	S40	—	—
SIRC	—	S40	S40	—	—
FXOSC	—	S40	S40	—	—
SXOSC	—	S40	S40	—	—
PLL	—	S40	S40	—	—
STCU	—	S40	—	—	—
WKPU	—	S40	S40	—	—
CMU	—	S40	—	—	—
PASS	—	S40	S40	—	—
TDM	—	S40	—	—	—
FCCU	—	S40	—	—	—
PMC_DIG	—	S40		—	—
LINFlex_1 to LINFlex_17	—	F40	—	—	F80

Table continues on the next page...

Table 9-6. Maximum clock frequencies - Peripherals (continued)

Module	HCLK	Peripheral CLK	BIU	Memory CLK	BAUD_CLK/Auxiliary CLK
LINFlex_0	—	F20	—	—	F40
DSPI_2, DSPI_3	—	S40	S40	—	F80
DSPI_0, DSPI_1	—	S40	S40	—	F40
SPI_0	—	S40	S40	—	F40
SPI_1, SPI_2	—	S40	S40	—	F80
SPI_3	—	S40	S40	—	F40
ADC_0,ADC_1	—	FS80	—	—	—
BCTU	—	FS80	FS80	—	—
eMIOS	—	FS80	FS80	—	—
IIC	—	F40	F40	—	—
FlexCAN	—	FS80	FS80	FS80	Refer module clock diagram for details
SAI	—	S40	S40	—	Refer module clock diagram for details
CRC	—	S40	—	—	—
RTC-API	—	—	S40	—	—
PIT-RTI	—	F40	F40	—	—
SWT0	—	—	S40	—	—
SWT1, SWT_HSM	—	—	S80	—	—
STM	—	—	FS80	—	—
DMAMUX_0, DMAMUX_1	—	FS80	FS80	—	—
NPC	—	—	S80	—	—
JTAGC	—	TCK (ext pin)	S40	—	—
CJTAG	—	TCK (ext pin)	S40	—	—
CMP_0 to CMP_3	—	S40	S40		—
MEMU_0	—	—	F80	—	—
MEMU_1	—	—	FS80	—	—
SIUL	—	F40	F40	—	—
LPU_CTL	—	FIRC	S40	—	—
INTC	—	—	FS80	—	—
SRAM (RAM256K)	—	—	S160	—	—
XBAR_LPU	—	—	S160	—	—

9.7.1 JTAG Frequencies

The following table shows the maximum frequencies of the JTAG interface.

Table 9-7. JTAG Frequencies

Configuration	e200z4 Max Internal JTAG Frequency	e200z2 Max Internal JTAG Frequency	Maximum External JTAG Frequency (TCK/TCK_ALT)	Notes
System Clock configured to 160 MHz	40 MHz	40 MHz	16 MHz	JTAG usable at 1/4 CPU frequency

9.7.2 Peripheral Clock Restrictions

There are some frequency requirements mandated by the protocol which should be considered while selecting DFS mode for an application.

- ENET AHB master clock interface should be greater than 50 MHz to support MII and RMII mode. ENET is sourced from clock node F80.
- ENET timer clock period should always be an integer. It is sourced from clock node F40 or FXOSC.
- BCLK which is bit clock of SAI and half of audio master clock should never be greater than BIU clock which is S40 as it will lead to underrun or overflow of FIFO inside SAI
- ADC conversion rate will drop as its frequency scales down. It is sourced from clock node FS80.
- FlexCAN protocol engine clock should always be less than its control host interface clock if MC_CGM_AC9_SC[SELCTL] is 0x0
- Flexray PE clock always need 80 Mhz clock (if FMPLL is clock source) or 40 Mhz clock (if FXOSC is clock source) to support 10/8/5/2.5 Mbit/s flexray data rates.

9.8 Default clock configuration

- **POWER UP:**After power is applied to the MPC5746C, the FIRC is the default clock for the system clock, and all clock sources and clock control blocks are reset to their default state. See clocking diagrams for default settings of dividers.
- **STANDBY/LPU EXIT:** System clock is based on configuration done in ME_DRUN_MC before LPU or STANDBY mode is requested. PLL can be selected as system source clock for faster low power exit time. Refer Accelerated Low Power Exit

9.9 Clock sources

This section provides the overview of the various clock sources.

9.9.1 PLL

The chip has a frequency modulated PLL with two outputs (PHI_0 and PHI_1). The separate system and peripheral clocks are provided with external dividers (see Figure 9-1 and Figure 9-15).

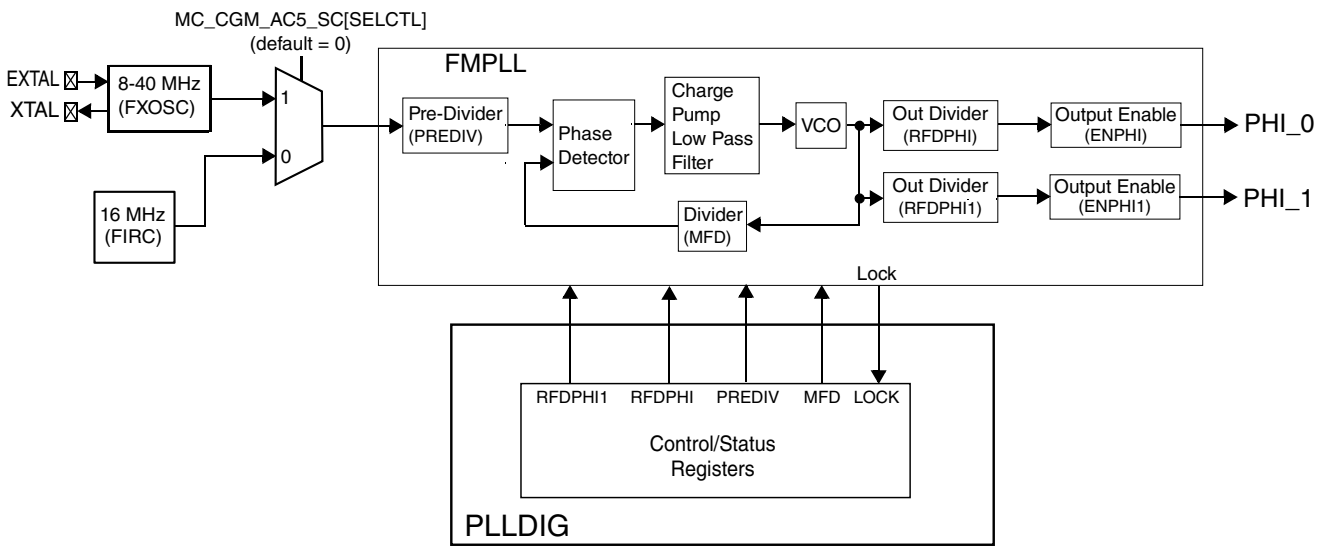


Figure 9-15. PLL block diagram

9.9.1.1 PLL input clocks

The clock source for the PLL can be either the external oscillator (FXOSC) or the 16 MHz internal oscillator (FIRC). The clock source is selected by configuring the MC_CGM_AC5_SC[SELCTL]. The input frequency range of the PLL is 8 – 40 MHz. The default input source for the PLL is the FIRC (MC_CGM_AC5_SC[SELCTL] = 0), but the FXOSC should be selected for best system clock accuracy (MC_CGM_AC5_SC[SELCTL] = 1).

9.9.1.2 PLL output clocks

The PLL has two output clocks:

- PHI_0
- PHI_1

PHI_0 is the primary output and is used to drive the system clock. It has a maximum frequency of 160 MHz. PHI_1 is the secondary output clock and has a maximum output frequency of 80 MHz.

NOTE

The PLL multiplication factor and output divider is software programmable.

9.9.1.3 PLL register interface

This device has a single digital interface for the PLL. The digital interface provides register control of all features of the PLL. Details are provided in the PLLDIG memory map and register definition section.

9.9.1.4 PLLDIG register reset values

All reset values for the PLLDIG registers that are device specific are shown in the following table. All other register reset values are shown in the PLL Digital Interface chapter.

Table 9-8. PLLDIG reset values

Register	Reset Value
PLL Calibration Register 3 (PLLDIG_PLLCAL3)	0000_0062h
PLLDIG PLL Control Register (PLLDIG_PLLCR)	0000_0400h
PLLDIG PLL Status Register (PLLDIG_PLLSR)	0000_0000h
PLLDIG PLL Divide Register (PLLDIG_PLLDV)	0C06_0018h

9.9.1.5 PLLDIG initialization information

Coming out of reset the PLL is disabled per the DRUN mode configuration register, MC_ME_DRUN_MC.

1. Configure PLL and related modules.

a. With PLL disabled, program PLL clock source in MC_CGM_AC5_SC[SELCTL]. Default is MC_CGM_AC5_SC[SELCTL] = 0 which selects the FIRC.

b. Program PLLDV[MFD] and PLLDV[RFDPHI].

c. If required, program the PLLFM register with the desired frequency modulation parameters and enable FM modulation, PLLFM[SSCGBYP] = 0. The PLLFM register must not be written after PLL enabled and operating in normal mode.

2. Turn on PLL.

a. Configure a mode configuration for turning on PLL, and keeping XOSC on. In a mode configuration register (for example, MC_ME_DRUN_MC), initialize MC_ME_<*mode*>_MC[XOSCON] = 1, MC_ME_<*mode*>_MC_PLLON] = 1. If desired, also set MC_ME_<*mode*>_MC[SYSCLK] needs to be initialized accordingly for this new mode configuration to use PLL as the sysclk (for example, MC_ME_RUN0_MC[SYSCLK] = 2 for PLL_PH0).

b. Enter that mode by two writes to MC_ME_MCTL register. First write should include the new mode and the key (use a single 32-bit write). The second write should include the same target mode but with the key inverted (again using a single 32-bit write). This is required even if entering the same mode.

3. Wait for the mode transition to complete.

a. Wait for the mode transition to complete by polling MC_MC_GS[MTRANS] or enabling an interrupt for flag MC_ME_IS[I_MTC]. A timer, even if it is the watchdog, should be used to make sure the mode transition completes. Mode transition will NOT complete until:

- XOSC counter expires (if the new mode configuration changes XOSCON to 1), and
- PLL is locked (if the new mode configuration changes MC_ME_<*mode*>_MC[PLLON] = 1)

b. Confirm the desired target mode was entered by checking the status of MC_ME_GS[S_CURRENT_MODE].

NOTE

See the "Mode Entry Module (MC_ME)" chapter in this *Reference Manual* more details.

9.9.2 Fast External Oscillator (FXOSC)

The external oscillator allows a crystal, or external clock, to be used as the reference for the MPC5746C. The input frequency range for the FXOSC is 8 – 40 MHz. The FXOSC clock source can be selected as the clock source for any operating mode (for example, DRUN, SAFE, and RUN[0:3]). The current operating mode is selected by configuring MC_ME_MCTL[TARGET_MODE]. Each of these operating modes can have the FXOSC as the clock source as selected in MC_ME_<*mode*>_MC for the corresponding operating mode. The FXOSC is the clock source for the RTI.

See Fast OSC Digital Interface chapter for details.

The Figure 9-16 shows the block diagram for FXOSC clock configuration.

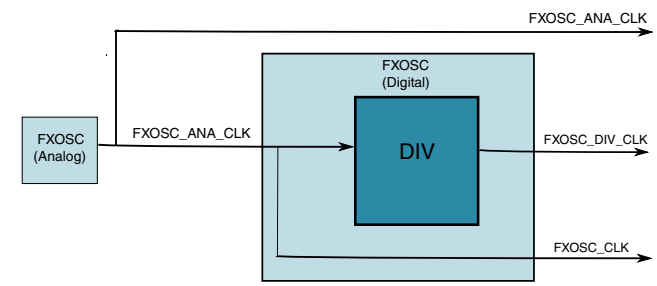


Figure 9-16. FXOSC Clock Configuration

9.9.2.1 FXOSC register reset values

All reset values for the FXOSC registers that are device specific are shown in the following table.

Table 9-9. FXOSC reset values

Register	Reset Value
FXOSC Control Register (FXOSC_CTL)	4030_0000h

NOTE

Reset value for I_OSC field depends on the DCF record for this field.

9.9.3 Slow external oscillator (SXOSC)

The SXOSC oscillator provides a 32 KHz clock with control and status registers accessible for application use (see Slow external oscillator for details).

The Figure 9-17 shows the block diagram for SXOSC clock configuration.

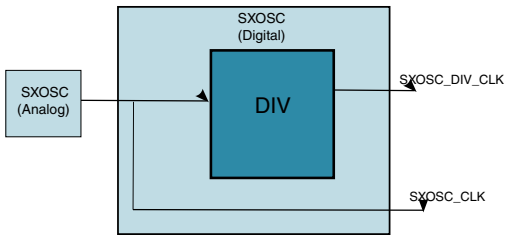


Figure 9-17. SXOSC Clock Configuration

9.9.3.1 SXOSC register reset values

All reset values for the SXOSC registers that are device specific are shown in the following table. All other register reset values are shown in the SXOSC chapter.

Table 9-10. FXOSC reset values

Register	Reset Value
SXOSC Control Register (SXOSC_CTL)	0040_0000h

9.9.4 16 MHz internal RC oscillator (FIRC)

The MPC5746C has a 16 MHz internal RC oscillator (FIRC) that is always enabled in RUN modes, and can be used as the clock source for the PLL. It is the clock source for the MC_RGM, FCCU, RTC, and SIUL input filters, and can also be used as a clock source for the ADCs and system clock. The FIRC is the default system clock after reset. The register interface is used for user trimming of the oscillator (see Internal 16 MHz RC Oscillator for details).

The Figure 9-18 shows the block diagram for FIRC clock configuration.

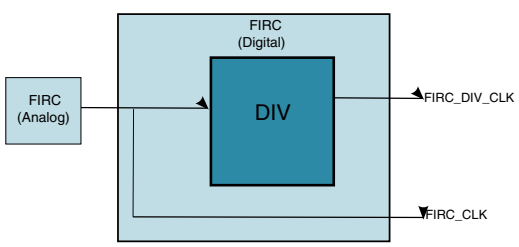


Figure 9-18. FIRC Clock Configuration

9.9.4.1 FIRC register reset values

The reset value of the FIRC Control register (FIRC_CTL) is device-specific and is shown in the following table.

Table 9-11. FIRC reset values

Register	Reset Value
FIRC Control Register (FIRC_CTL)	00*nn*_0011h

NOTE

Bits 10-15 are trim bits, with values determined during factory test to ensure the most accurate clock timing. To maintain accurate timing, these bits should never be written by application software.

9.9.5 128 kHz internal RC (SIRC)

This internal RC oscillator provides the 128 kHz clock with high accuracy, low power, fast stabilization, and fast recovery (see Internal 128 kHz RC oscillator chapter for details).

The Figure 9-19 shows the block diagram for SIRC clock configuration.

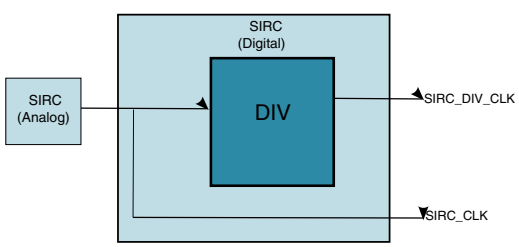


Figure 9-19. SIRC Clock Configuration

9.9.5.1 SIRC register reset values

The reset value of the SIRC Control register (SIRC_CTL) is device-specific and is shown in the following table.

Table 9-12. SIRC reset values

Register	Reset Value
SIRC Control Register (SIRC_CTL)	00*nn*_0011h

NOTE

Bits 8-15 are trim bits, with values determined during factory test to ensure the most accurate clock timing. To maintain accurate timing, these bits should never be written by application software.

9.10 Clock monitoring

The device supports clock monitoring of PLL and FXOSC clock. FIRC is used as reference clock (see Clock Monitor Unit (CMU) and CMU configuration for details).

9.10.1 CMU configuration

This section explains the CMU configuration.

Figure 9-20 shows the block diagram for the CMU on the MPC5746C.

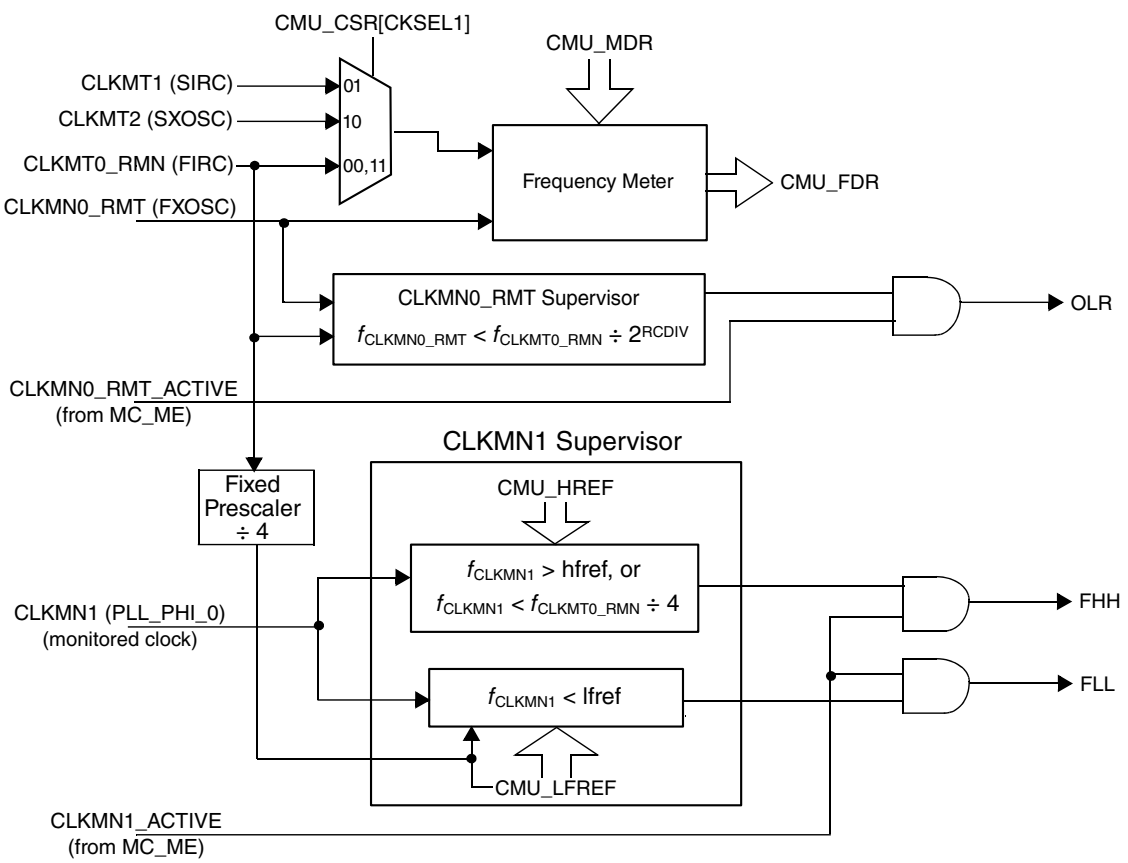


Figure 9-20. CMU block diagram

9.10.1.1 Clock input sources

The following table shows the clocks that are monitored by the CMU. These signals are connected internally on the chip, but are not accessible via pins on the boundary of the device. FIRC is the reference clock for clock monitor.

Table 9-13. Clock input sources

Clock module	Monitored clock
CMU	FXOSC, FIRC

9.10.1.2 PLL Monitor

Software can program an upper and lower limit for the expected PLL PHI_0 output clock frequency. If the monitor is enabled, and the measured frequency is above or below the limits, a flag bit is set in a status register. This will lead to FCCU_NCF[32] fault being set when the flag bit is set. The default condition of the clock monitor is disabled (CMU_CSR[CME] = 0). The FIRC is used as the reference clock for the monitor (CLKMT0_RMN, and CMU_CSR[CKSEL1] = 00b or CMU_CSR[CKSEL1] = 11b).

9.10.1.3 FXOSC monitor

The frequency of the FXOSC is compared to a minimum limit as configured in the CMU. If the measured FXOSC frequency is below the limit, a flag bit is set in a status register (CMU_ISR[OLRI]). The OLR signal from the CMU is one of the functional reset sources of MC_RGM, and an interrupt could be generated (depending on MC_RGM configuration, see 'Functional' Event Reset DisableAlternate Request Status Register (MC_RGM_FEAR)). It is not recommended to configure OLR signal from CMU as interrupt if FXOSC is selected as system clock.

If CMU_ISR[OLRI] is set, then status in CMU_ISR and in RGM_FES would be cleared by the software routine. However on such a condition, till the time FXOSC is not corrected, the CMU_ISR[OLRI] would not recur again. Hence the software should ensure that FXOSC is corrected by polling of MC_ME_GS[FXOSC_ON]. Also, the software should not attempt any mode transition until ME_GS[FXOSC_ON] is set. Otherwise the mode transition would hang.

OLR will not be captured in STANDBY mode as CMU is power gated. When MCU is in Standby mode and counters are working on FXOSC clock and FXOSC is running slow then expected wake-up can be delayed or prevented without notification. For example, if

crystal clock dies during STANDBY mode and crystal clock is configured to be ON in target DRUN mode, then mode transition to DRUN will not complete. However SWT can be kept running on SIRC and can generate wake-up on timeout to protect against such cases when FXOSC is not functioning correctly. Additionally, external wakeups can also always cause a low power exit.

NOTE

The FXOSC monitor is always enabled when the FXOSC and FIRC are valid (see field descriptions MC_ME_GS[S_FXOSC] and MC_ME_GS[S_FIRCON] in the Global Status Register (MC_ME_GS)).

9.10.1.4 FIRC monitor

The frequency of the 16 MHz FIRC (as well as SIRC and SXOSC) can be calibrated with the CMU using the FXOSC as a reference (CLKMN0_RMT (FXOSC)). This allows application trimming of the FIRC frequency. The application specifies the duration of the measurement period in units of FIRC clock periods. The clock monitor provides in a register the number of references clocks counted during the measurement period, which can be used to calculate the actual FIRC frequency (see Frequency meter for details).

9.10.1.5 System Clock Monitors

Clock monitoring at the device level is not required for the HSM, as it contains its own internal monitor.

9.10.1.6 CMU register reset values

All reset values for the CMU registers that are device specific are shown in the following table. All other register reset values are shown in the CMUchapter.

Table 9-14. CMU reset values

Register	Reset Value
CMU Control Status Register (CMU_CSR)	0000_0006h
CMU Interrupt Status Register (CMU_ISR)	0000_0000h

9.11 Loss of system clock behavior

This device has built-in mechanisms for detecting loss of the oscillator or PLL clocks, and provides several options for reaction to a loss of clock in the application. See Figure 9-21.

9.11.1 Loss of PLL lock/FXOSC clock

As shown in Figure 9-21, each loss of lock signal from the PLL and the FXOSC failure signals are monitored by the FCCU. The FCCU can be programmed to generate a short or long reset sequence when a clock failure occurs, or simply to generate an interrupt. By default, FCCU is configured to generate Long reset sequence when a clock failure occurs.There is no automatic system clock switch, and the user is required to program the switch through the Mode Entry module (MC_ME).

If a long reset is selected the PLL, FXOSC, MC_ME, and MC_CGM are reset to their default states, and the system clock is switched to the FIRC. If a short reset is selected, the clock configuration is maintained, but again, the PLL backup clock is the only available clock to clock the system through an MC_ME mode switch.

NOTE

If PLL lock is lost and PLL clock dies immediately, FCCU will generate reset but status will not be captured as its BIU_CLK was not available. This can be debugged if MC_RGM_FES indicates FCCU as source of reset and no status reported in FCCU.

NOTE

If FXOSC failure happens, FXOSC registers will reset to its default state. It is recommended to check S_FXOSC bit of Global Status Register (MC_ME_GS) before using FXOSC clock source again.

9.11.2 Loss of FIRC clock

The frequency of the FIRC clock is monitored by the frequency meter in CMU. There is no automated trigger of an FCCU error condition if the FIRC fails. Since the FIRC is the boot and backup clock, a failure is a catastrophic failure.

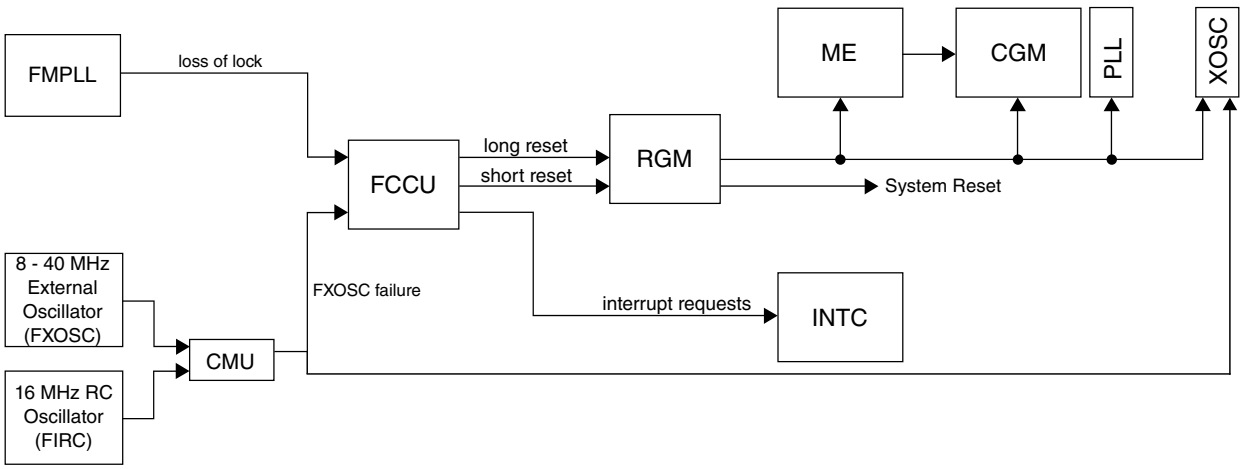


Figure 9-21. Loss of clock logic

Chapter 10
Embedded Memories

10.1 Overview

The embedded memory architecture for devices includes:

- Onboard SRAM, including system SRAM
- Onboard system error correction code (ECC) flash memory
- End-to-end ECC (e2eECC) error detection and correction
- Built-in flash memory security features including censorship and a programmable Hardware Security Module (HSM)
- Embedded memories in the peripherals[1] such as, FlexCAN, eDMA, FlexRay, and ENET.

10.2 System SRAM

MPC5746C devices include 512 KB of general-purpose on-chip ECC SRAM, configurable for either zero- or one-wait state read operation latency.

For power-sensitive applications it is important to note which ranges of SRAM are available in each device run mode. See Table 10-1 for overview of SRAM run mode availability. See the Power Management chapter Introduction for details.

Table 10-1. SRAM run modes

Start Address	End Address	Allocated Size [KB]	Run Mode(s)
0x40000000	0x40001FFF	8 (STANDBY)	All
0x40002000	0x4000FFFF	56	RUN, STOP. Optionally STANDBY, LPU [1]
0x40010000	0x4001FFFF	64	RUN, STOP. Optionally STANDBY, LPU[1]

Table continues on the next page...

1. Refer to the respective module chapter for details.

Table 10-1. SRAM run modes (continued)

Start Address	End Address	Allocated Size [KB]	Run Mode(s)
0x40020000	0x4003FFFF	128	RUN, STOP. Optionally STANDBY, LPU[1]
0x40040000	0x4007FFFF	256	RUN, STOP

1. Can be independently configured to be available during STANDBY or LPU modes.

NOTE

It is recommended to use SMPU to protect the SRAM cuts which one intends to put in SLEEP mode using the GPR Sleep Memory register.

10.2.1 SRAM Controller (PRAMC)

Each of the microcontroller's 2 banks of system SRAM has a dedicated Platform RAM Controller (PRAMC) module. The modules appear in the Peripheral Bridge A memory map as:

- PRAMC_0
- PRAMC_1

Each PRAMC module controls access to its associated SRAM by bus masters.

- PRAMC_0 corresponds to SRAM address space 0x4000_0000 to 0x4003_FFFF.
- PRAMC_1 corresponds to SRAM address space 0x4004_0000 to 0x4007_FFFF.

Additionally, the PRAMC_0 module is integrated with the flash controller (PFLASH) module to provide overlay SRAM support.

NOTE

1. PRAMC configuration settings are not global, i.e., each PRAMC module's settings affect only the SRAM array associated with that module.
2. Each SRAM array can be configured for either zero- or one-wait state read operation latency via settings in its associated PRAMC module's PFCR1 register.
3. If the system frequency is set to 160 MHz, then the PRAMCx_PRCR1[FT_DIS] field should be set to 1.
4. See the RAM controller chapter for details on configuring the SRAM controller.
6. See System memory map for the MPC5746C SRAM map.

10.2.1.1 SRAM controller (PRAMC) registers

Not all of the PRAMC register fields shown in the RAM Controller (PRAMC) chapter are applicable to all PRAMC modules. The following sections show the applicable register fields for each PRAMC module.

10.2.1.1.1 PRAMC_0 valid control register bits

Figure 10-1. PRAMC_0 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PRAMC_0_PRCR1	R	0															
	W																
	R	0						PRI		P1_BO_DIS	P0_BO_DIS	0					FT_DIS
	W																

10.2.1.1.2 PRAMC_1 valid control register bits

Figure 10-2. PRAMC_1 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
PRAMC_1_PRCR1	R	0															
	W																
	R	0									P0_BO_DIS	0					FT_DIS
	W																

10.2.2 Overlay SRAM

The MPC5746C includes the capability of mapping a portion of system SRAM over specific regions of on-chip flash memory so that any access to an overlaid flash address is routed to the overlay SRAM instead. This enables the calibration of constant data without requiring additional external RAMs and calibration memory interfaces.

Note

Overlay SRAM is normally used for system development purposes only. Execution from overlay RAM remapped over Flash can be enabled or disabled by use of PFCRCR[IRMEM].

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

1. Using PFlash Calibration Region Descriptors (PFCRDn) define one or more (up to 8) overlay regions.
2. Enable individual regions using the appropriate bits in the PFLASH Remap Descriptor Enable Register (PFDCRDE).
3. Enable overlay using the PFCRCR[GRMEN] field.

NOTE

On the microcontroller, the only overlay RAM target available is system RAM array0 (0x40000000 – 0x4003FFFF). There are no restrictions within that range.

See PFlash calibration remap support for details on memory overlay (remapping).

10.3 Flash memory

Flash memory on MPC5746C devices consists of a flash memory controller and a flash memory array module. The flash controller provides flash configuration and control functions and manages the interface between the flash memory array and the device crossbar switch. The following figure shows the memory architecture.

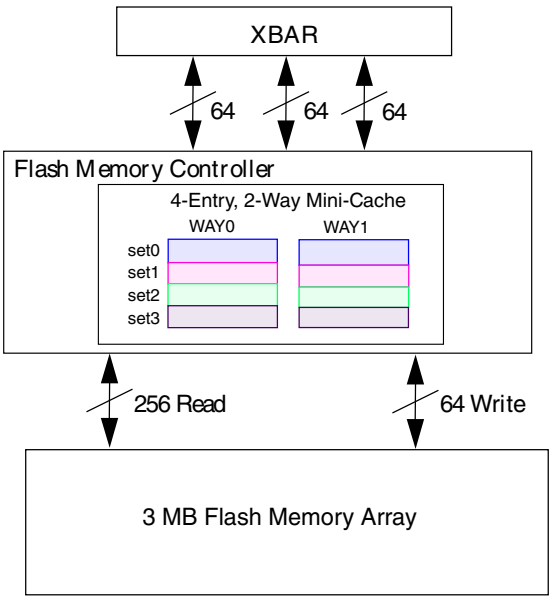


Figure 10-3. MPC5746C flash memory block diagram

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

10.3.1 Flash memory controller

The MPC5746C flash memory controller acts as an interface between the system bus and the flash memory array and serves as the interface to the on-chip system SRAM for calibration remapping.

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

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

See Flash memory controller for details.

NOTE

PFLASH_PFCR1[APC] and PFLASH_PFCR1[RWSC] field values needs to be updated while read accesses to flash are stopped, that is, the software code which updates the PFLASH_PFCR1[APC] and PFLASH_PFCR1[RWSC] fields must be executed from RAM.

High clock speeds require the PFLASH_PFCR1[RWSC] field setting to a non-zero value to insert one or more wait states into flash accesses. For more information, refer MPC5746C data sheet. Operating with RWSC and APC settings outside of those specified in the data sheet is not supported and may result in corrupted read data.

10.3.2 Flash memory array

MPC5746C devices include a single 3 MB flash memory array, referred to as "main space," plus an independent 16 KB block of one-time-programmable (OTP) flash memory included to support systems that require non-volatile memory for security features or system initialization information. The independent 16 KB block is referred to as "UTEST space."

Reads of the embedded flash memory return a 256-bit page of data that may be buffered in the mini-cache in the flash memory controller. Programming of the flash may be done by double word (64-bits), page (256-bits), or quad-page (1024-bits). Flash memory is erased on a block by block basis.

NOTE

Entry to FLASH low power mode while a program/erase operation is ongoing should be prohibited, or this can result in unpredictable behavior.

10.3.2.1 Features

- Flash segmentation
  - Two 16 KB, four 32 KB, and two 64 KB, and ten 256 KB blocks provided for third party code
  - Eight 16 KB provided for boot code EEPROM emulation
  - Two 64 KB and one 16 KB blocks provided for secure code

    - Two 16 KB blocks provided for secure data
    - Support for reading-while-writing when the accesses are to different partitions
- Flash memory protection: write protection and OTP function available for each block
- Test and initialization data stored in a non-volatile 16 KB OTP UTest block
- ECC with double bit detection, single bit correction
- Erase suspend, program suspend, and erase-suspended program all supported

10.3.3 Flash organization

The embedded flash memory consists of blocks from 16 KB to 256 KB, divided into partitions that determine locations for valid read-while-write (RWW) operations. While the embedded flash memory is performing a "write" (program or erase) to a given partition, it can simultaneously read from any other partition.

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

The following figure shows the flash block segmentation and read-while-write partitioning for devices.

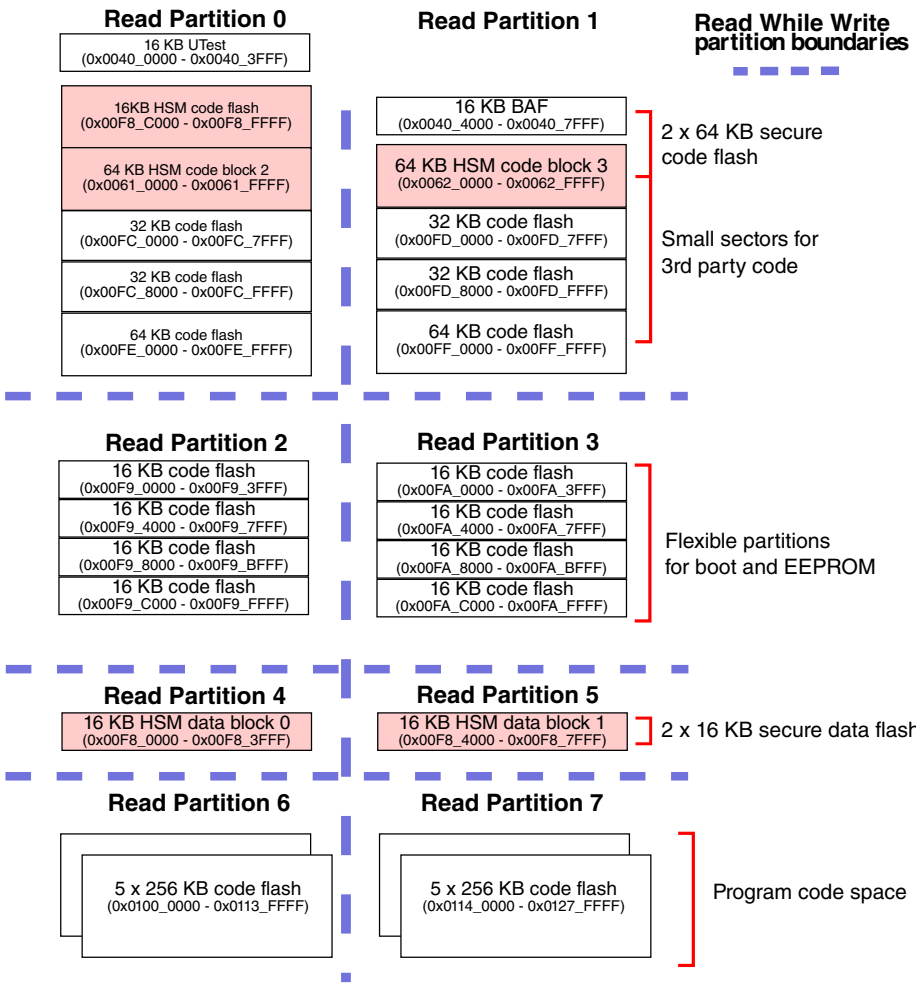


Figure 10-4. Flash memory segmentation and read-while-write partitioning

Note

See Table 1 for the flash memory map.

10.3.4 UTest NVM memory space

Devices in the family contain a 16KB area of One-Time Programmable (OTP) flash memory for storing test information and device configuration data. Most of the device configuration data is in the form of Device Configuration Format (DCF) records.

- See Introduction for details on DCF record structure, function, and contents.
- See Table 1 for the UTest flash memory map.

10.3.5 Overlay flash memory

In addition to the ability to use on-chip system RAM for overlay, the microcontroller can also reroute accesses to on-chip flash memory to other addresses within on-chip flash memory. The mechanism and controls are similar, with the following exceptions:

- Accesses to special-purpose flash blocks, i.e., UTEST flash, EEPROM, secure flash, and BAF cannot be rerouted, nor can accesses to other blocks be rerouted to the UTEST flash, EEPROM, secure flash, or BAF blocks.
- When specifying addresses for overlay flash, i.e., the flash memory addresses that accesses are to be rerouted to, the *mirrored* flash address must be used. The mirrored address range refers to the normal range (0x00F8C000 – 0x127FFFF), with address field bit[27][2] is set, giving a range of 0x08F8C0000 - 0x0927FFFF.

See PFlash calibration remap support, for details on memory overlay (remapping).

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

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

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

2. where the least significant bit of the address field is numbered '0'

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

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

A comparison of the traditional and e2eECC approaches is shown in the following table.

Table 10-2. ECC comparison of data write-then-read sequence

Traditional ECC	End-to-End ECC (e2eECC)
Bus master initiates data write	Bus master initiates data write and generates ECC checkbits based on 29-bit address and 64-bit data fields for the computational shell or 32-bit data field for the peripheral shell
Data write transfer routed from bus master to appropriate bus slave	Data write transfer (including checkbits) routed from bus master to appropriate bus slave
For a bus memory slave, generate the ECC checkbits based on the data value and store data + checkbits into the memory	For a bus memory slave, store data and checkbits into the memory
Bus master initiates data read of previously written memory location	Bus master initiates data read of previously written memory location
Data read transfer routed from bus master to appropriate bus slave	Data read transfer routed from bus master to appropriate bus slave
For a bus memory slave, the memory array is accessed, the controller performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and drives the read data onto the system bus interconnect	For a bus memory slave, the memory array is accessed, and the controller passes the read data and associated checkbits onto the system bus interconnection

Table continues on the next page...

Table 10-2. ECC comparison of data write-then-read sequence (continued)

Traditional ECC	End-to-End ECC (e2eECC)
The bus master captures the read data and continues	The bus master captures the read data and associated checkbits, performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and continues

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

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

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

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

10.5 Security features

The MPC5746C architecture uses a combination of device life cycle monitoring and password protection to limit the conditions under which debugging interfaces can access certain areas of flash memory. For more information, refer, Security Reference Manual.

Chapter 11
Device Configuration Format (DCF) records

11.1 Introduction

Device Configuration Format (DCF) records are used to configure certain registers in the device during system boot while the reset signal is asserted. An individual DCF record consists of a pointer to the location of a register internal to the device and the data to be written to that register.

There are two broad categories of DCF records, TEST DCF Records and UTEST DCF Records. TEST DCF records are developed by the factory and are used mainly to program registers involved in trimming trip points for voltage comparators, adjusting analog to digital voltage supplies, trimming oscillator frequencies, and enabling RAM repair. However these records can also be used to write any DCF client. The TEST DCF Records are programmed into TEST flash during production and cannot be modified. TEST flash is not visible to the user.

The second set of DCF records are UTEST DCF records. Some UTEST DCF records are written by the factory and programmed during production testing (see DCF_Clients spreadsheet attached to this document). Others are written by the end user and programmed at the same time application code is programmed into the flash memory. User-supplied UTEST DCF records start at the next location in the UTEST memory map following the UTEST DCF records programmed into the flash memory during factory production. The user-defined DCF records set up the initial memory map, define which tests the Self-Test Control Unit (STCU2) will run during the boot sequence, enable the HSM, assign memory blocks as OTP, assign flash blocks to be associated with specific password groups and assign flash memory blocks to specific tamper detect regions.

System boot is a complex process requiring a considerable amount of initialization to take place before releasing reset. Before the device can be used in an application, the user application code, reset vectors for all of the CPUs and the DCF records must be properly programmed into their respective flash memories.

When power is applied to a properly programmed device, the Power Management Controller (PMC) takes control of the device. After the system power supplies have reached predefined levels, the Power Management Controller signals the Reset Generation Module (RGM) to begin the boot sequence. During the boot sequence, the RGM enables another internal module, the System Status and Control Module (SSCM), to read the device configuration records and then write the configuration information to the specified registers. After registers are initialized, the SSCM starts the Self-Test Control Unit (STCU2), which begins running the tasks enabled by the DCF record information.

11.2 DCF clients

DCF clients are 32-bit wide hardware registers inside a module, that receive and store the data from a DCF record. This stored data is used to initialize registers and to configure features. DCF Clients have a default value before any DCF Records are written, and may have special writing constraints, such as, ‘Write Once’ or allowing bits to be written from ‘1’ to ‘0’ only or vice versa. DCF clients need not implement all 32 bits. DCF clients maybe designated ‘TEST DCF record only’, which means that only DCF Records stored in TEST flash can write to the DCF Client. The DCF Records select the ‘target’ DCF client by a 30-bit field in the DCF Record consisting of a 15-bit ‘Chip Select’ field and a 15-bit Address field. Each module that includes DCF clients is assigned a Chip Select during chip definition. The address field is only relevant to the address decoding within that module and may not necessarily relate to the address of a register visible to software.

11.2.1 Safety features of DCF clients

Depending on its role in the device a client may be equipped with a safety feature or a combination of safety features.

11.2.1.1 Parity

If a DCF client implements parity checking, then it will receive a parity bit additionally to it's data in the DCF record. During device operation it will continuously monitor whether the data it stores matches the parity and report errors to the SSCM.

11.2.1.2 Triple voting

DCF clients that use the triple voting have three copies of the register. The SSCM will write to all three registers in a single write cycle. During device operation the DCF client continuously monitors whether all three copies match and reports errors to the SSCM. The device will use the majority result, so single errors will not affect device operation.

11.2.1.3 Spreaded triple voting

DCF clients that use spreaded triple voting have three copies of the register which are individually addressed and where the data is stored in three different formats. The SSCM will fetch these records independently. During device operation the DCF client continuously monitors whether all three copies can be resolved from the stored format to the value and reports errors to the SSCM. The device will use the majority result, so single errors will not affect device operation.

11.2.1.4 Second write check

This feature is used by DCF clients which receive their records twice - e.g. once before and once after BIST for clients which retain their value during self test. They will monitor whether the second write matches the first and report deviations to the SSCM.

11.2.2 DCF client modification rules

Depending on its role in the device a client may implement one or a combination of modification rules. If a modification rule is in effect, the order in which DCF records are placed in the record list may be important.

11.2.2.1 Write Once

A DCF client using the Write Once rule can only be written with a single DCF record. Records appended later in the list will be ignored and not change the value of the client.

11.2.2.2 Write 0 only

A bit in a DCF client can only be written from a 1 to a 0. So, if an earlier DCF record has set a bit to 0, then an attempt by a later record to set this bit to a 1 will be ignored.

11.2.2.3 Write 1 only

A bit in a DCF client can only be written from a 0 to a 1. So, if an earlier DCF record has set a bit to 1, then an attempt by a later record to set this bit to a 0 will be ignored.

11.3 DCF record structure

A DCF record is a double-word (64-bit) entry consisting of the following:

- Control word: Information to locate the corresponding DCF client internal to the device (pointer to the location of a register internal to the device)
- Data word: Data to be written to that client

DCF Records select the target DCF client by a 30-bit field in the DCF Record consisting of a 15-bit ‘Chip Select’ field and a 15-bit Address field. Each module that includes DCF clients is assigned a Chip Select during chip definition. The address field is only relevant to the address decoding within that module and may not necessarily relate to the address of a register visible to software.

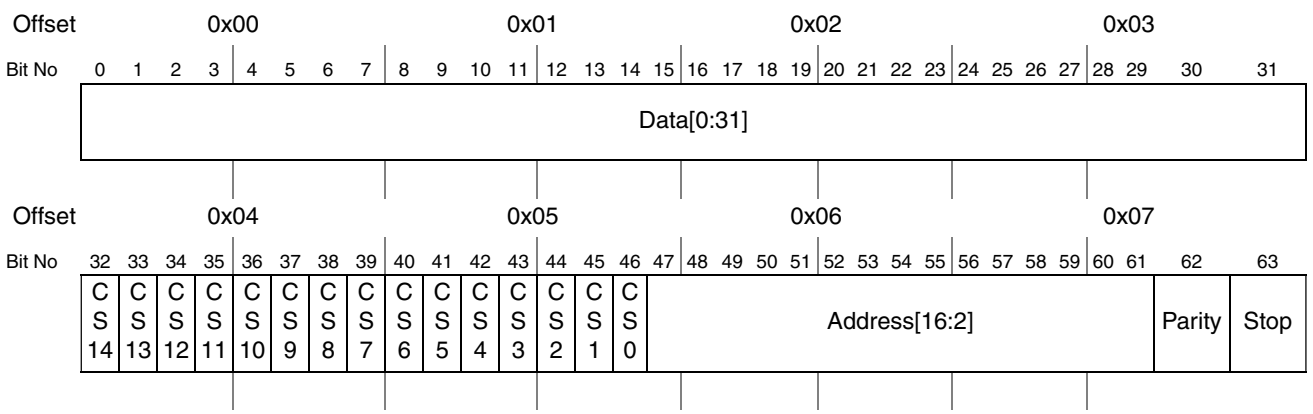


Figure 11-1. DCF Record Structure

Table 11-1. DCF Record Field Description

Field	Name	Description
0-31	Data[0:31]	32 bits of data that is to be written to the DCF Client
32-46	CS*n*	Chip Select *n* One Chip select is asserted (‘1’) per DCF record to select the target module for the DCF client. All other Chip Selects should be negated (‘0’).
47-61	Address[16:2]	Address of the DCF client within the selected module.

Table continues on the next page...

Table 11-1. DCF Record Field Description (continued)

Field	Name	Description
		Note: Address decoding for DCF clients may not match the standard software address map decoding. Details of DCF Client addresses are defined in each module chapter.
62	Parity	Parity Bit for the DCF Record. Note: This bit is NOT implemented for DCF Client written from UTEST.
63	Stop	Stop bit. Indicates the end of the list of DCF Records. Stop = ‘0’ - NOT the end of the list Stop = ‘1’ - End of the list Note: The Erased state of flash is 0xFFFF_FFFF_FFFF_FFFF. Therefore the list ends with the first unprogrammed double word. This location can be programmed with a new record to extend the list.

11.4 DCF records sequence

An individual DCF record consists of information to locate the corresponding DCF client internal to the device (control word) and the data to be written to that client (data word).

DCF records appear as contiguous series of entries programmed at a specific address within flash memory. The following structure must be present:

- The first DCF record must be a start record. This record must be placed at the beginning of a DCF area in flash memory to indicate to the device that the following records must be processed.
- DCF records containing configuration data must immediately follow the start record with no blank records between-an unprogrammed record is interpreted as a stop record and no DCF records following that record are processed. This allows one to program the records in several sessions, each time appending new records at the end of the list.
- The end of the configuration records are indicated by the presence of a stop record; indicated by the stop bit being set. Normally, the stop bit is never set in a valid UTEST DCF record programmed during production, since this would prevent appending additional UTEST records. The flash memory location following the last UTEST DCF record programmed at the factory is an unprogrammed location. Therefore, that location's contents will be 0xFFFF_FFFF. Thus, the stop bit location in this unprogrammed flash memory location will be a logic 1, signifying that this is the last DCF record and it is not to be acted upon.

SSCM recognizes the following record as a start record.

Table 11-2. DCF start record

0x00 (0:31)	0x04 (32:63)
0x05AA_55AF	0x0000_0000

The start bit is set by the factory at the beginning of the UTEST flash memory area.

SSCM recognizes the following record as a stop record.

Table 11-3. DCF stop record

0:31	32:62	63
Ignored	Ignored	1

User-supplied DCF records may be added in a contiguous manner immediately following the factory-written DCF records. There must never be an unprogrammed record in the series of DCF records as it will be interpreted as a stop record.

The following table shows the series of DCF records when *n* data records are stored in the UTEST flash.

Table 11-4. Series of DCF records in UTEST flash memory

Record type	ADDR offset	DATA			
Start record	0x00	0x05AA_55AF			
	0x04	0x0000_0000			STOP=0
Data records	0x08	WDATA[31:0]			
	0x0C	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
	0x10	WDATA[31:0]			
	0x14	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
	...	...			
Stop record	8(*n*-1) + 0x0	Reserved			
	8(*n*-1) + 0x4	Reserved			1
	8*n* + 0x0				
	8*n* + 0x4				

It is possible for more than one DCF Record to write to the same DCF client. In this case the later record usually overrides a DCF client value set by a previous record. However, not all DCF clients allow overwrites; this depends on the DCF client implementation.

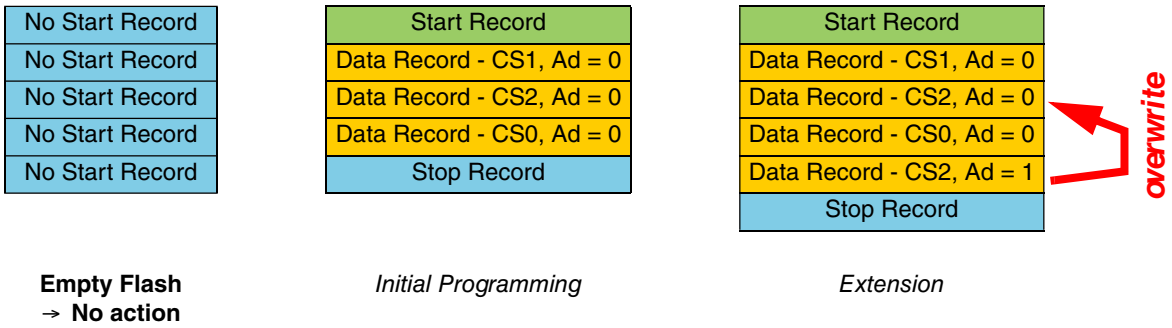


Figure 11-2. Appending DCF records

11.5 Chip configuration records

The DCF Clients Table contains information on DCF clients available in the chip. See the UTEST-FULL DCF excel attached to this document.

11.5.1 External Regulation Exit Trim Wait (PMCDIG_ERETW)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		DELAY_VAL[2:15]													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DELAY_VAL[16:31]															
W																
Reset	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0

The following table describes the register functions.

Table 11-5. PMCDIG_ERETW register field descriptions

Field	Description
0-1 Reserved	This field is reserved. This read-only field is reserved and always has the value zero.
2-15 DELAY_VAL[2:15]	Programmable External Regulation Trim Delay Value This can be programmed according to the external regulator stablization delay time. This value will hold no significance where external regulator startup time is fast or external regulator is turned off during low power modes.

Table continues on the next page...

Table 11-5. PMCDIG_ERETW register field descriptions (continued)

Field	Description
16-31 DELAY_VAL[16:31]	Programmable External Regulation Trim Delay Value This can be programmed according to the external regulator stablization delay time. This value will hold no significance where external regulator startup time is fast or external regulator is turned off during low power modes.

11.5.2 Security and serial boot mode records

For a description for the Flash backdoor password, serial boot mode and security records, please refer to the SSCM chapter.

11.5.3 STCU configuration records

The STCU provides a large number of configuration records. Please see, Self-Test Control Unit (STCU2) for more details.

Chapter 12
Debug Overview

12.1 Introduction

The device contains multiple Nexus clients (Nexus class 3+) that communicate over an auxiliary port interface compliant to IEEE-ISTO 5001-2012 standard and two sets of JTAG ports compliant with IEEE 1149.1 standard and 1149.7 cJTAG standard.

Nexus class 3+ module provides real-time development capabilities for e200z2 and e200z4 core processor in compliance with Class 3 of the IEEE-ISTO 5001-2012 standard, with additional Class 4 features available.

The debug requirements covers instruction and data trace across the 2 host cores, instruction trace from the instruction trace of Z0 core inside the HSM and data trace of accesses to system RAM. In addition, compact CJTAG is supported alongside standard JTAGC.

NOTE

Throughout this manual, the e200Z4204n3 core is referred as Z4 or e200Z4.

Similarly, the e200z210n3 core is referred as Z2 or e200Z2.

The debug solution is created by the following modules:

- JTAGC - IEEE-1149.1 operations
- CJTAGC for IEEE 1149.7 operations (note full 4-pin JTAG operation is the default mode of operation)
- JDC - required to support the Challenge Response password authentication protocol via the HSM module
- e200z43 nexus class 3 module
- e200z2 nexus class 3 module
- Nexus Data Trace client (NXMC) for the SRAM.
- Nexus 2 trace client for Z0 core of HSM
- Nexus Port Controller, capable of supporting 16xMDO full port mode and 12xMDO reduced port mode

The communication with Nexus Debug Interface (NDI) is handled via the auxiliary port and the JTAG port.

- The auxiliary port comprises 20 output pins and one input pin. The output pins include one message clock out (MCKO) pin, 16 message data out (MDO) pins, two message start/end out (MSEO0, MSEO1) pins, and one event out (EVTO) pin. Event in (EVTI) is the only input pin for the auxiliary port.
- There are two sets of JTAG ports, each consists of three inputs and one output. Only BGA324 supports two sets of JTAG ports.These pins include test data input (TDI), test data output (TDO), test mode select (TMS), and test clock input (TCK). TDI, TDO, TMS, and TCK are compliant with the IEEE 1149.1-2001 standard and are shared with the NDI through the test access port (TAP) interface. Ownership of the TAP is achieved by loading the appropriate enable instruction for the desired Nexus client in the JTAG controller (JTAGC). See Memory map and registers section for the JTAGC opcodes to access the different Nexus clients.

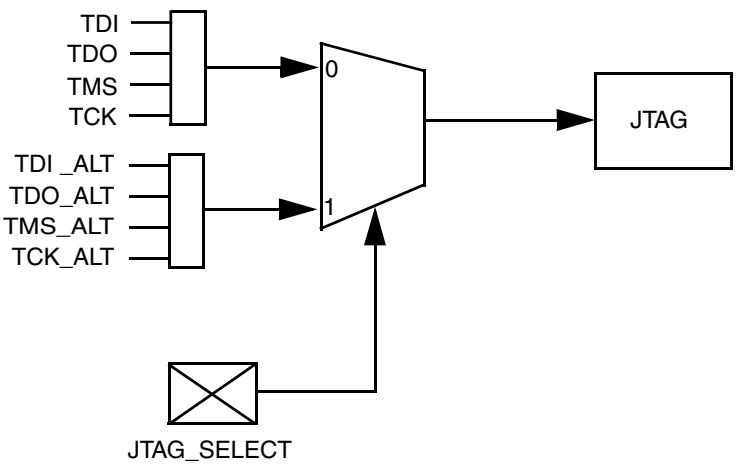


Figure 12-1. JTAG Ports

NOTE

For details on JTAG pads, see the device I/O Signal Description and Input Multiplexing tables attached to this document as an excel file.

The following peripherals support debug features:

- STM_0, STM_1
- SWT_0, SWT_1, and HSM_SWT
- FCCU
- eMIOS_0, eMIOS_1
- SAI0, SAI1, and SAI2
- ENET
- IIC_0, IIC_1, IIC_2, and IIC_3
- DSPI[3:0]

- FLEXCAN[7:0]
- eDMA controller
- SPI[3:0]
- PIT-RTI
- RTC-API
- SSCM

NOTE

Debugger connection would break after STANDBY exit if Pad Keeper functionality is enabled in STANDBY modes.

12.2 Features

The debug module of the device compliant with the IEEE-ISTO 5001-2012 standard. The following features are implemented:

- 29-bit pin interface for high throughput, including two sets of four JTAG pins
  - Two modes are supported: full port mode (FPM) and reduced port mode (RPM). FPM comprises 16 MDO pins. RPM comprises 12 MDO pins, and can be used to increase the number of GPIOs. Care must be taken as bandwidth will be limited.
  - At ambient temperature, the Nexus Aux port MCKO supports the maximum frequency of 80 MHz both at FPM and RPM. Otherwise, the maximum supported frequency is reduced to 40 MHz.
  - Auxiliary output port
    - One MCKO (Message clock out) pin
    - 16 MDO (Message data out) pins
    - Two MSEO (Message start/end out) pins
    - One EVTO (Event out) pin
  - Auxiliary input port uses one EVTI (Event in) pin.
  - Two sets of JTAG ports each uses four pins (TDI, TDO, TMS, TCK, TDI_ALT, TDO_ALT, TMS_ALT, and TCK_ALT)
- The NPC block performs the following functions:
  - Control arbitration for ownership of Nexus Auxiliary Output Port between all the nexus clients
  - Nexus Device Identification Register and Messaging
  - Generates MCKO enable and frequency division control signals
  - Controls sharing of EVTO between different nexus clients.
  - Generates an MCKO clock gating control signal to enable gating of MCKO when the auxiliary output port is idle.
  - Control of the device-wide debug mode
  - Generates asynchronous reset signal for Nexus blocks

- Host processor (e200z4) and secondary processor (e200z2) development support features:
  - IEEE-ISTO 5001-2012 standard class 3+ compliant.
    - Program trace via Branch Trace Messaging (BTM), with the option of using Branch history messaging to enhance message throughput.
    - Data trace via data write messaging (DWM) and data read messaging (DRM). This allows the development tool to trace reads and/or writes to selected internal memory resources.
    - Ownership trace via ownership trace messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An ownership trace message is transmitted when a new process/task is activated, allowing development tools to trace ownership flow.
    - Run-time access to the on-chip memory map via the JTAG port.
    - Watchpoint messaging (WPM) via the auxiliary port. This allows a watchpoint to be set, which then sends a watchpoint message each time the watchpoint is hit. Unlike watchpoint triggering, WPM does not stop the core or start trace.
    - Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint messaging.
  - Additional class 4 features
    - Watchpoint trigger enable of program and/or data trace messaging. This is an extension of WPM to allow a watchpoint to stop or start program or data trace.
    - Processor overrun control
- Cross triggering — The capability for an EVTO (event out) signal from either the e200z4 or e200z2 Nexus3+ to generate a debug request to the other core, thus allowing both cores to enter debug mode within a short period of each other.

12.3 Debug scheme

The debug scheme and connectivity is shown in the following figure:

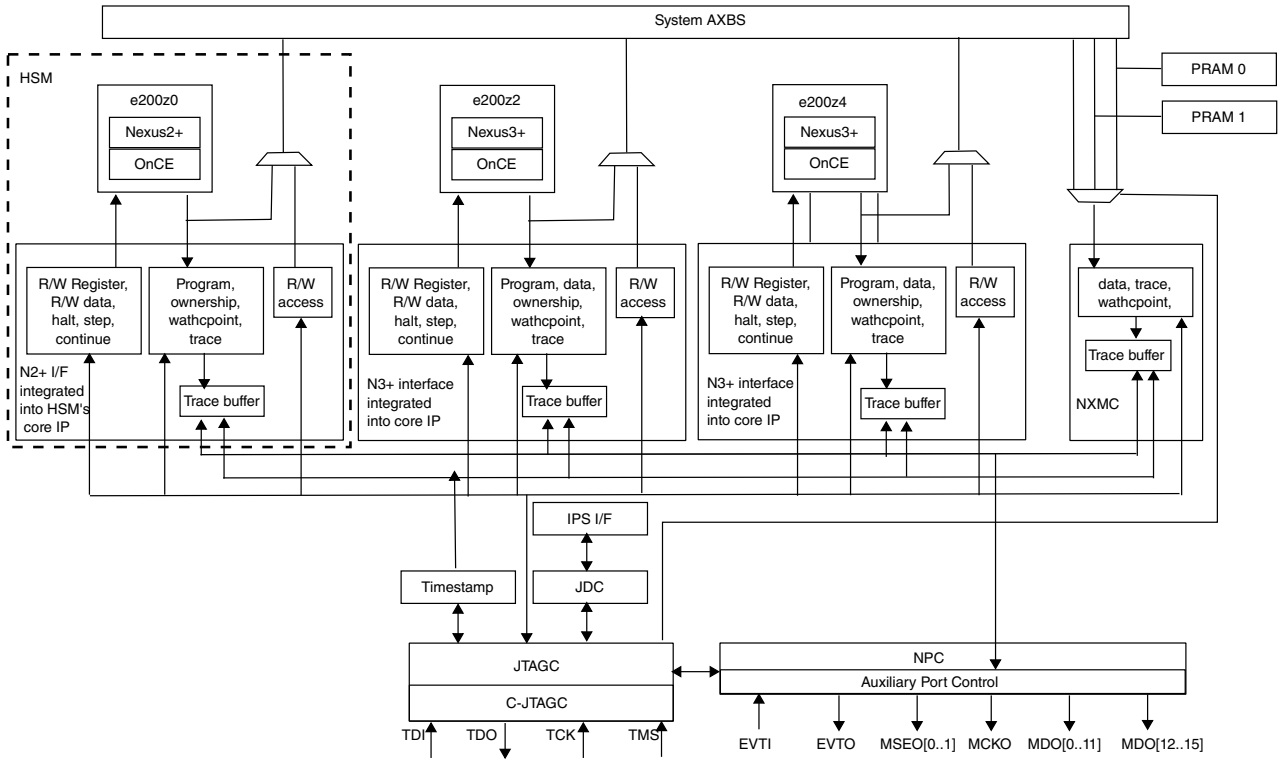


Figure 12-2. Debug scheme

12.4 Modes of operation

The Nexug debug interface (NDI) block is in reset when the TAP controller state machine is in the TEST-LOGIC-RESET state. The TEST-LOGIC-RESET state is entered on the assertion of the power-on reset signal, or through state machine transitions controlled by TMS. Ownership of the TAP is achieved by loading the appropriate enable instruction for the desired Nexus client in the JTAG controller (JTAGC) block. The NPC transitions out of the reset state immediately following negation of power-on reset.

12.4.1 Nexus Reset Mode

In Nexus reset mode, the following actions occur:

- Register values default back to their reset values.
- The message queues are marked as empty.
- The auxiliary output port pins are negated if the NDI controls the pads.
- The TDO output buffer is disabled if the NDI has control of the TAP.

- The TDI, TMS, and TCK inputs are ignored.
- The NDI block indicates to the MCU that it is not using the auxiliary output port. This indication can be used to three-state the output pins or use them for another function.

12.4.2 Full-port Mode

In the full-port mode (FPM), 16 MDO pins (MDO[0..15]) pins are used to transmit Nexus messages. All trace features are available by writing to the configuration registers via the JTAG port. FPM is entered by asserting the MCKO_EN and FPM bits in the Port Configuration Register (PCR).

12.4.3 Reduced-port Mode

In the reduced-port mode (RPM), the number of MDO pins is reduced from 16 to 12 (MDO[0..11]). All trace features are available by writing to the configuration registers via JTAG. However, the FIFO overflow errors are more likely to occur in this mode, especially if the data trace is enabled. RPM is entered by asserting the MCKO_EN bit and negating the FPM bit in the Port Configuration Register (PCR).

12.4.4 Disabled-Port Mode

In the disabled-port mode, auxiliary output pin port enable signals are negated, thereby disabling the message transmission. Any debug feature that generates messages cannot be used. The primary features available are class 1 features and read/write access to the registers. Class 1 features include the ability to trigger a breakpoint event indication through EVTO. The NDI is placed in disabled mode upon exit of power-on reset.

12.4.5 Multi mode nexus TAP

In this mode, Nexus Debug interface of e200Z4 and e200Z2 gets connected in a daisy fashion. This allows the tools to treat each core individually and then start (or stop) all cores together with a longer scan chain. Note that the HSM is not included in this daisy chain.

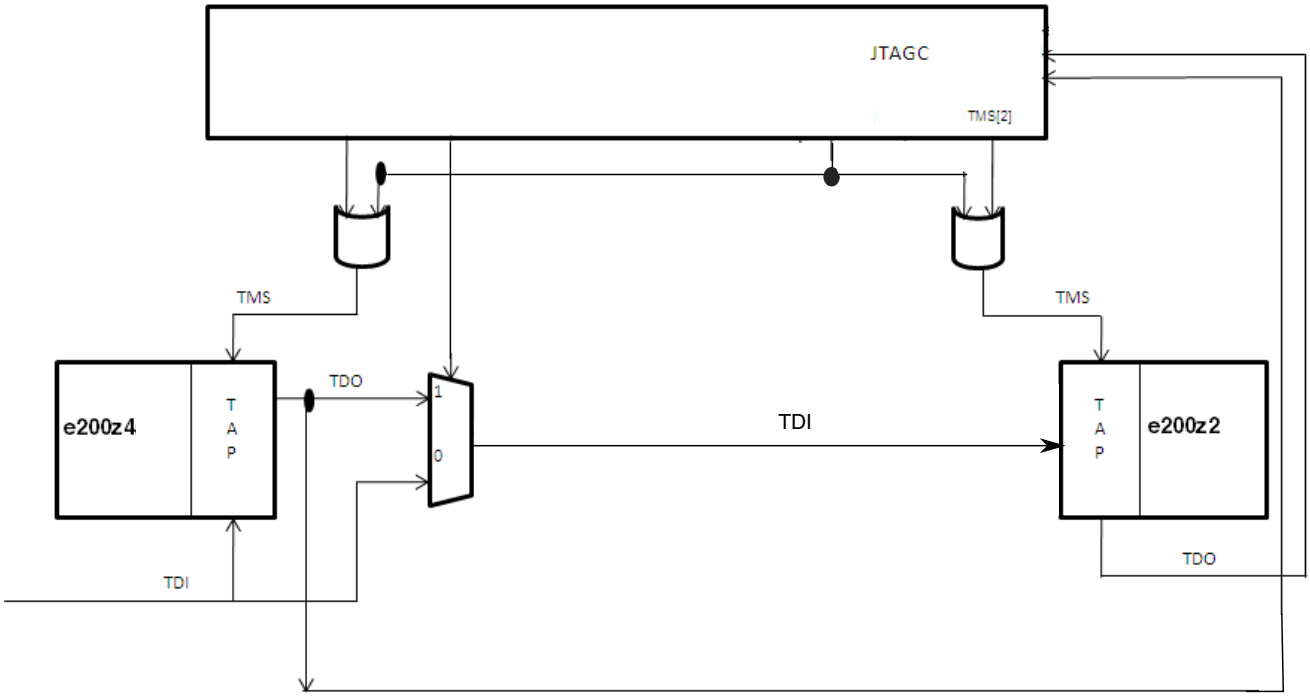


Figure 12-3. Multi mode nexus TAP

12.5 External signal description

The auxiliary and JTAG pin interfaces provide for the transmission of messages from Nexus modules to the external development tools and for access to Nexus client registers. The auxiliary/JTAG pin definitions are outlined in the following table:

Table 12-1. Auxiliary/JTAG pin definitions

Name	Port	Function
EVTO	Auxiliary	Event Out pin
EVTI	Auxiliary	Event In pin
MCKO	Auxiliary	Message Clock Out pin (from NPC)
MDO[0:15]	Auxiliary	Message Data Out pins
MSEO[0:1]	Auxiliary	Message Start/End Out pins
TCK	JTAG	Test Clock Input
TDI	JTAG	Test Data Input
TDO	JTAG	Test Data Output
TMS	JTAG	Test Mode Select Input

12.6 Memory Map and Registers

The NDI block contains no memory mapped registers. Nexus registers are accessed by the development tool via the JTAG port using a client select and a register index. The client select is controlled by loading the correct access instruction into the JTAG controller. After the desired client TAP is selected, OnCE registers for that core client are accessible by loading the appropriate value in the RS[0:6] field of the OnCE command register (OCMD). Other Nexus clients (data trace client on SRAM) use a Nexus access command to access the Nexus registers of the clients. Nexus is enabled, and the associated Nexus registers become accessible, by loading the NEXUS_ENABLE instruction into the RS[0:6] field of the OCMD. When Nexus register access is enabled, the desired Nexus registers are accessible.

12.7 Functional Description

The block is implemented by integrating the following blocks:

- Nexus Port Controller Block
- Dual Nexus e200z4 Development Interface (OnCE and Nexus3+ subblocks)
- Nexus e200z2 Development Interface (OnCE and Nexus3+ subblocks)
- NPC_HNDSHK module
- NXMC module

NOTE

The TAP controller logic, reset logic, and some miscellaneous logic are duplicated in all these blocks.

12.7.1 Enabling Nexus Clients for TAP Access

Once the NDI is out of the reset state, the loading of a specific instruction in the JTAG controller (JTAGC) block is required to grant the NDI ownership of the TAP. Each Nexus client has its own JTAGC instruction opcode for ownership of the TAP, granting that client the means to read/write its registers. Once the JTAGC opcode for a client has been loaded, the client is enabled by loading its NEXUS-ENABLE instruction. Opcodes for all other instructions supported by Nexus clients can be found in the relevant sections of this chapter.

12.7.2 TAP Sharing

Each of the individual Nexus blocks on the MCU implements a TAP controller for accessing its registers. The JTAGC controls the ownership of the TAP so that the interface to all of these individual TAP controllers appears to be a single port from outside the device. Once a Nexus client has been granted ownership of the TAP, any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive.

12.7.3 Configuring the NDI for Nexus Messaging

The NDI is placed in disabled mode upon exit of power-on reset. If message transmission via the auxiliary port is desired, a write to the port configuration register (PCR) located in the NPC is then required to enable the NDI and select the mode of operation. Asserting MCKO_EN in the PCR places the NDI in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting the FPM bit selects full-port mode.

When writing to the PCR, the PCR LSB must be written to a logic 0. Setting the LSB of the PCR enables factory debug mode and prevents the transmission of Nexus messages.

The following table describes the NDI configuration options:

Table 12-2. NDI Configuration Options

MCKO_EN bit of PCR	FPM bit of PCR	Configuration
0	X	Disabled
1	1	Full-Port Mode
1	0	Reduced Port Mode

12.7.4 TCK clock relation with NPC clock

The TCK frequency should always be less than or equal to half of the NPC clock frequency.

12.7.5 Programmable MCKO frequency

MCKO is an output clock to the development tools used for the timing of MSEO and MDO pin functions. MCKO is derived from the system clock, and its frequency is determined by the value of the MCKO_DIV field in the port configuration register (PCR) located in the NPC. Possible operating frequencies include NPC_CLK, one-half, one-quarter, and one-eighth NPC_CLK speed.

The following table shows the MCKO_DIV encodings. In this table, NPC_CLK represents the system clock frequency of 80 MHz.

Table 12-3. MCKO_DIV values

MCKO_DIV[2:0]	MCKO Frequency
0b000[1]	NPC_CLK
0b001	NPC_CLK/2
0b010	Reserved
0b011	NPC_CLK/4
0b100	Reserved
0b101	Reserved
0b110	Reserved
0b111	NPC_CLK/8

1. The NPC_CLK setting for MCKO frequency should only be used if this setting does not violate the maximum operating frequency of the auxiliary port pins (80 MHz).

NOTE

The Nexus Port Controller Port Configuration Register MCKO Divider bits (NPC_PCR[MCKO_DIV]) can be set to 0b000 to select a 1X clock rate as the Nexus Auxiliary output port frequency for the MCKO and MDO pins. Depending on the system frequency, this may force the MCKO and MDO pins to switch at a frequency higher than can be supported by the pins. This maximum frequency is specified in the device electrical specification of the Nexus MCKO and MDO pins. Insure that the maximum operating frequency of the MDO and MCKO pins is not violated when setting the NPC_PCR[MCKO_DIV] values.

The following table shows the NPC clock frequencies across DFS modes.

Table 12-4. NPC clock frequencies

Use cases	when HSM is enabled	when HSM is disabled
Full Run	80	80

Table continues on the next page...

Table 12-4. NPC clock frequencies (continued)

Use cases	when HSM is enabled	when HSM is disabled
LDFS80	40	40
NLDFS80	40	80
NLDFS40	20	40
NLDFS20	10	20

12.7.6 Cross Triggering Control

To enable a debug event in one core to cause a debug event in the other core at approximately the same time, the EVTO signal from the e200z2 Nexus3+ or e200z4 Nexus3+ is connected to the other core's DEVT1 input. When enabled in each core's Nexus1 DBCR0 register, a pulse of the DEVT1 signal causes a debug event to occur. In this case, only one external EVTO signal is generated and each core controls whether or not EVTO causes a debug event to occur.

Cross triggering of breakpoints between cores is supported. As a minimum this should include the ability to stop and start cores synchronously. i.e., break on one core can stop other cores immediately and run on one core can trigger other cores to run. Cross triggering with the HSM is NOT to be supported since it is to be completely independent.

Interconnection of debug mode control signals are shown in the following figure.

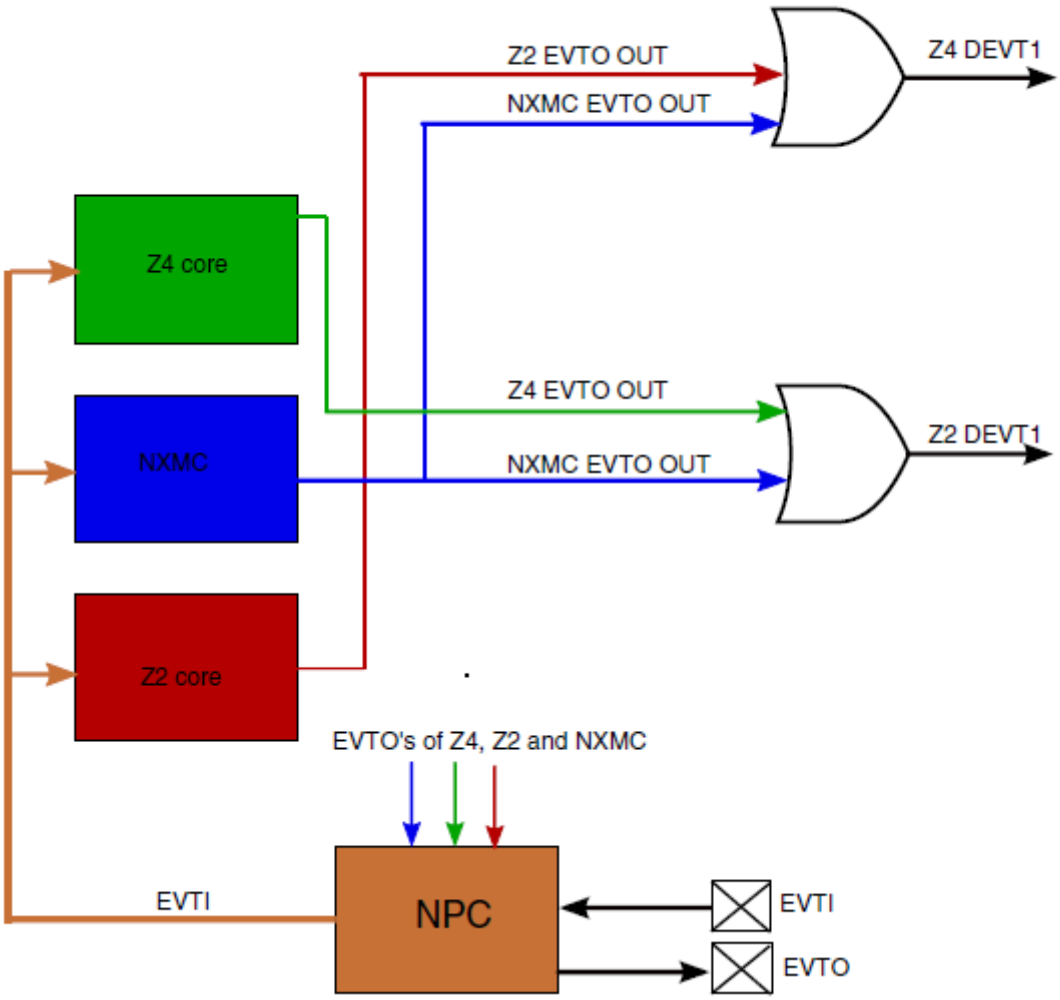


Figure 12-4. Debug Mode Control Interconnections

12.7.7 Time Stamping

The device implements Nexus trace time-stamping support logic module. This provides a mechanism to insert time-stamping data into the trace stream from each nexus client. Enable and disable controls of this time-stamping feature is included in the JTAGC, using some of the available auxiliary data bits of DATA1 register.

The time-stamping should run on the fastest core clock frequency.

- The time-stamping is based upon the full 30-bit time-stamp.
- The time-stamp is required to be reset only upon a power on reset.

12.8 Nexus Port Controller (NPC)

The Nexus Port Controller (NPC) is that part of the NDI that controls access and arbitration of the device's internal Nexus modules. The NPC contains the port configuration register (PCR) and the device identification register (DID).

The device has two instances of NPC. NPC_0 is responsible for the traces and interaction with pads and remains in full power domain. NPC_1 exists in always on domain and interacts with NPC Handshake (NPC HNDSHK). When the system enters into the low power mode, the NPC_1 and the NPC HNDSHK together follow a protocol handshaking with the debugger. The handshake enables the load and store of the context appropriately by the debugger.

NOTE

NPC_0 TAP should be reset on LPU_RUN to DRUN transition if nexus tracing is required to be enabled in DRUN mode.

The protocol to be followed by the debugger for each low power mode is described in the following section.

For details, see the NPC chapter.

12.8.1 NPC Handshake module

This module together with the NPC_1 enables debug entry/exit across the low power modes. The NPC Handshake supports:

- setting and clearing of the PCR sync bit in NPC_1 on low-power mode entry and exit
- putting the core into debug mode on low-power mode exit
- generating a falling edge on the JTAG TDO pad on low-power mode exit

Entry into STOP/STANDBY from RUN modes or Entry into LPU_STOP/ LPU_STANDBY from LPU_RUN mode:

- For the device to be in debug mode on low power exit, the debugger should set the LP_DBG and LP1_SYNC bit in the NPC_1 PCR register.
- On STOP, STANDBY, LPU_STOP or LPU_STANDBY mode entry, the LPU controller asserts the LP mode entry request input after it gets an indication from MC_ME that the clock disable process has completed and before the processor enters its stopped state. The mode transition will then not proceed until the LP mode entry acknowledge output has been asserted to the LPU controller.
- The notification to the debugger of a low-power mode entry consists of setting the NPC_1 PCR [LP1_SYNC] bit by the NPC Handshake module.

- The debugger acknowledges that the transition into a low-power mode may proceed by clearing NPC_1 PCR [LP1_SYNC] bit.
- The device enters into the low power modes. In anticipation of the low-power mode exit notification, the TDO pad is driven to '1'.

Exit from STOP/STANDBY into RUN modes or Exit from LPU_STOP/ LPU_STANDBY into LPU_RUN mode:

- On STOP or LPU_STOP mode exit, the LPU controller asserts the LP mode exit request input after ensuring that the regulator and memories are in normal mode and before the processor exits its stopped state. The mode transition will then not proceed until the LP mode exit request has been acknowledged. The MC_RGM asserts the exit from standby input when executing a reset sequence due to a STANDBY or LPU_STANDBY exit. The reset sequence will then not complete until the LP mode exit request has been acknowledged.
- The notification to the debugger of a low-power mode exit consists of driving the TDO pad to '0'.
- The debugger acknowledges that the transition from a low-power mode can continue by setting the NPC_1 PCR [LP1_SYNC] bit.
- Further, the NPC_1 PCR[LP1_sync] bit is de-asserted by the hardware and the LP mode exit acknowledge output is sent to the LPU controller. The chip comes out of low power mode with the core in the debug mode.

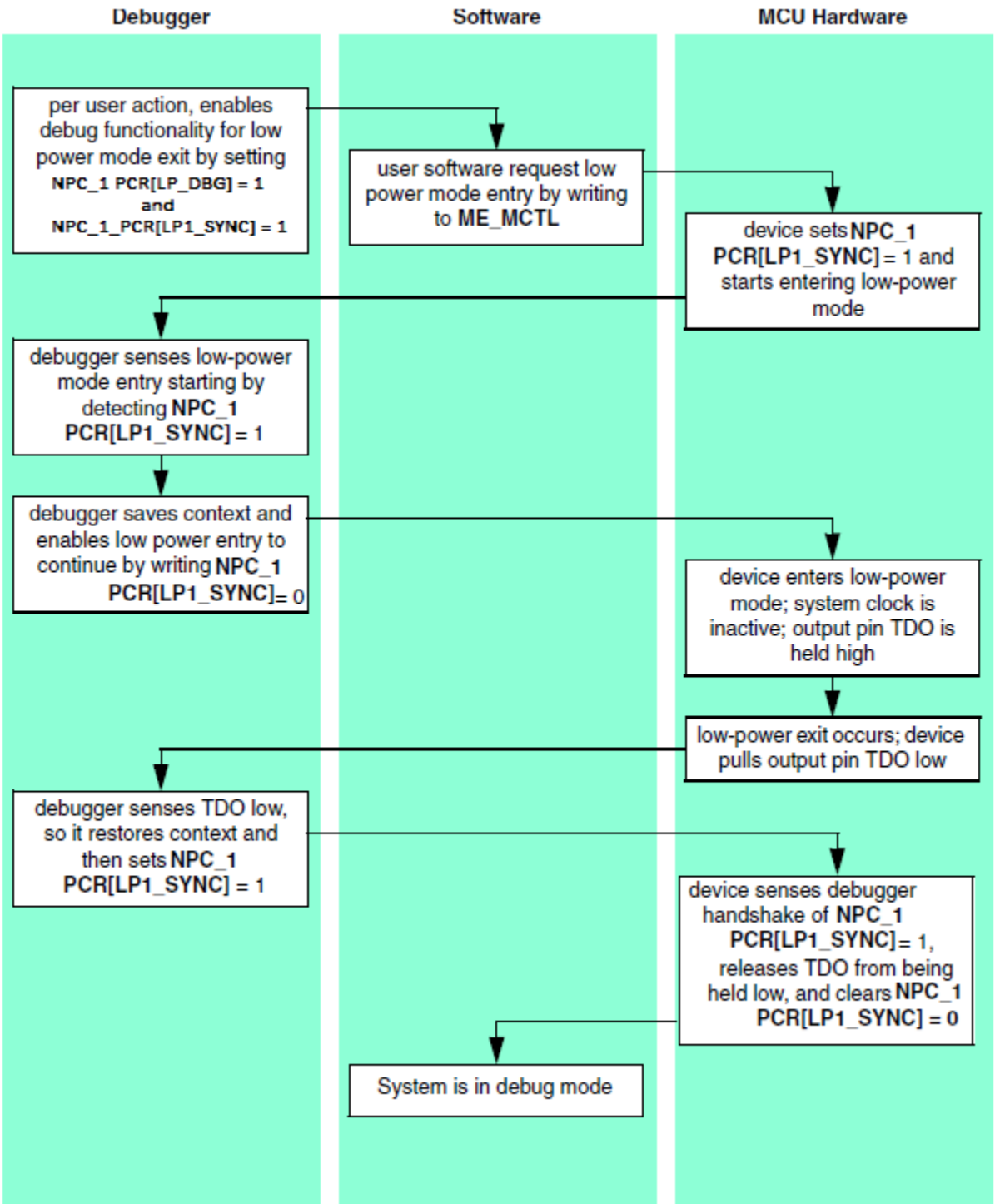


Figure 12-5. Low-Power mode debug handshake flow Ddiagram for STOP, STANDBY, LPU_STOP, LPU_STANDBY

Entry into LPU_RUN mode from RUN modes:

- For the device to be in debug mode on LPU_RUN mode entry, the debugger should set the NPC_1 PCR [LP_DBG] and NPC_1_PCR[LP1_SYNC].

- On LPU_RUN mode entry, the LPU controller asserts the LP mode entry request input after it gets an indication from MC_ME that the clock disable process has completed and before the processor enters its stopped state. The mode transition will then not proceed until the LP mode entry acknowledgement has been asserted to the LPU controller.
- The notification to the debugger of a low-power mode entry consists of setting the NPC_1 PCR[LP1_SYNC] bit.
- The debugger acknowledges that the transition into a low-power mode may proceed by clearing the NPC_1 PCR[LP1_SYNC].
- If the target mode is LPU_RUN and not the stop/standby modes, the debugger should observe '0' on TDO pad and acknowledge that the transition into LPU_RUN mode can continue by setting the NPC_1 PCR[LP1_SYNC].
- Further, the NPC_1 PCR[LP1_sync] bit is de-asserted by the hardware and the LPU RUN entry acknowledgement is sent to the LPU controller. The device comes into LPU_RUN mode with the Z2 core in the debug mode.

For getting Z2 core out of debug mode, see the Core debug support chapter.

Exit from LPU_RUN mode into RUN modes:

- For the device to be in debug mode on RUN mode entry, the debugger should set the NPC_1 PCR[LP_DBG] and NPC_1_PCR[LP1_SYNC].
- On LPU_RUN mode exit, the LPU controller asserts the LP mode entry request after it gets an indication from MC_ME that the clock disable process has completed and before the processor enters its stopped state. The mode transition will then not proceed until the LP mode entry acknowledgement has been asserted to the LPU controller.
- The notification to the debugger consists of setting the low-power mode handshake bit in the NPC_1 PCR[LP1_SYNC].
- The debugger acknowledges by clearing the NPC_1 PCR[LP1_SYNC].
- If the target mode is RUN and not the LPU stop/standby modes, the debugger should observe '0' on TDO pad and further acknowledge that the transition into RUN mode can continue by setting the NPC_1 PCR[LP1_sync] bit.
- Further, the NPC_1 PCR[LP1_sync] bit is de-asserted by the hardware and the LP mode exit acknowledgement is sent to the LPU controller. The device comes into RUN mode with the Z4 core in the debug mode.

For getting Z4 core out of debug mode, see the Z4 core debug support chapter.

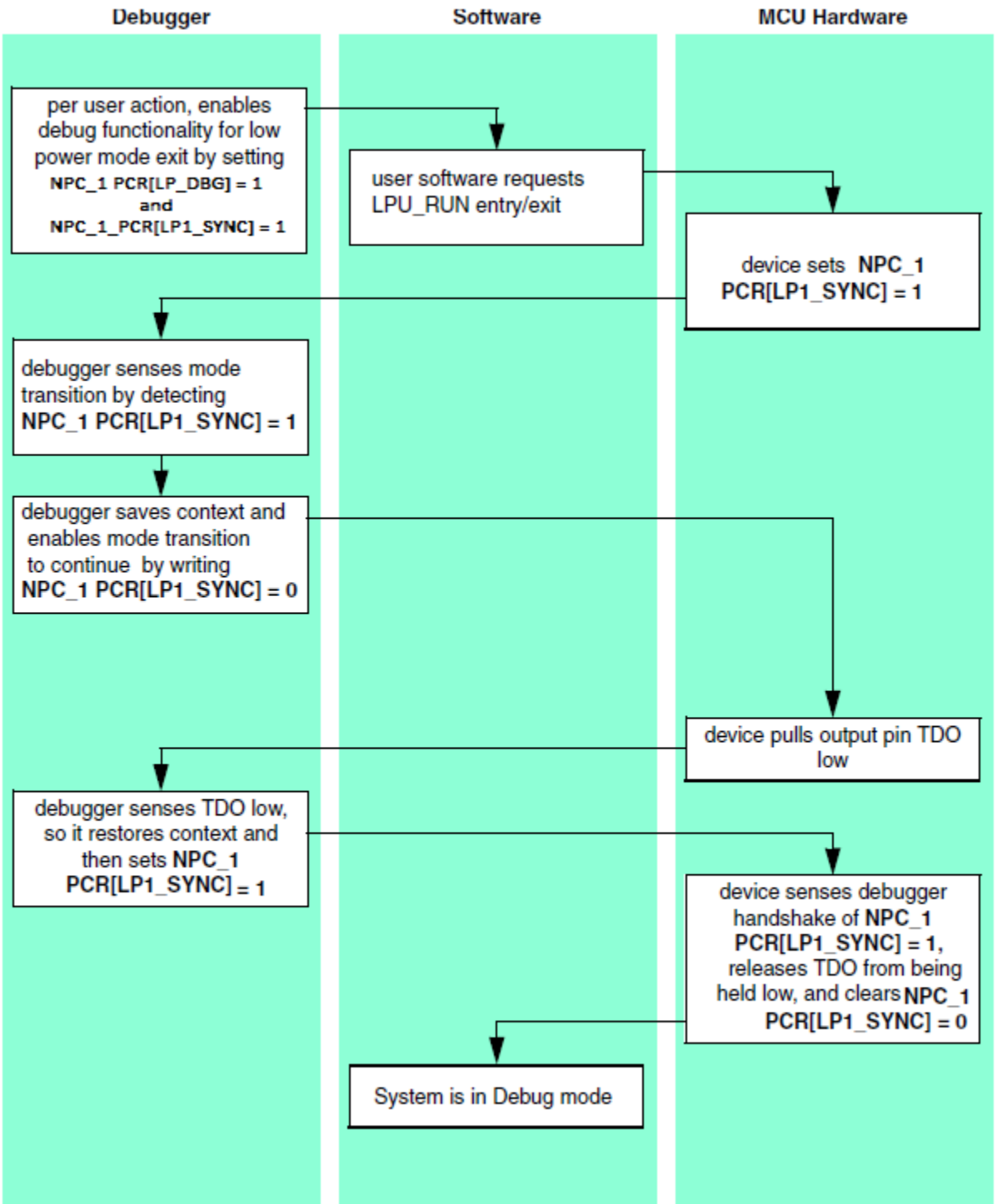


Figure 12-6. Low-Power mode debug handshake flow diagram for LPU_RUN entry and exit

12.9 e200z4 core and e200z2 core debug support

Internal debug support in the e200z4 core and e200z2 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes.

For details, see the e200z4 Core Debug Support chapter and e200z2 Core Debug Support chapter.

12.10 JTAG Controller (JTAGC)

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format. All Nexus debug registers are accessed and configured via the JTAG controller.

Sharing of TAP with other TAP controllers via ACCESS_AUX_x instruction (nexus clients/ other TAPs for debug placed as aux TAPs).

AUX port selection: When ACCESS_AUX_x instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive.

DATA 1 register: It is used for SRAM_MUX support, timestamping support and for disabling port enables from NPC to aux pads. See JTAGC Control 1 register.

For details, see the JTAGC chapter.

12.11 JTAG ID

The JTAG ID of the device is dependent on SIUL2_MIDR1[MAJOR_MASK] and SIUL2_MIDR1[MINOR_MASK] register field value.

SIUL2_MIDR 1[MAJOR_MASK]	SIUL2_MIDR 1[MINOR_MASK]	Version Number	Design Center	PIN	Manufacturer ID	Constant	JTAG_ID
		4 bit	6 bit	10 bit	11 bit	1 bit	32 bit
0x0	0x1	0x1	0x26	0x84	0xE	0x1	1988401D
0x1	0x1	0x1	0x26	0x85	0xE	0x1	1988501D

12.12 CJTAGC

The CJTAG is based on IEEE 1149.7. It has two wire interface (TCK and TMS) with default mode as 4-wire interface. Switches to 2-wire with standard switching sequence. For details, see Switching from Standard Protocol to Advanced Protocol.

Unused ports can be assigned for other purposes once switching is done. No special configuration required for the device. For details, see CJTAG chapter.

12.13 JDC

The JDC allows movement of data from IPS to JTAG domain registers (and back), but it does not perform the top level JTAG functions such as aux port sharing or boundary scan register control. It is currently used to allow communication between CPU and external tools for things such as security password challenge/response.

For details, see JDC chapter.

12.14 Debug implementation

This section describes the practical implementation of the debug port, its management, and the multiplexing strategy of its pads. It is required on the device to support the Class 3+ (2012) debug features on the e200Z4 and minimum Class 3 (2012) on the e200Z2.

12.14.1 Debug Capabilities

The device supports the Class 3 and Class 3+ debug features listed in the following table.

Table 12-5. Nexus3 Debug Requirements

Feature	Class 1	Class 2	Class 3	Class 4
Static debug	X	X	X	X
Set breakpoints/watchpoints	X	X	X	X
Watchpoint messaging	X	X	X	X
Ownership trace	-	X	X	X
Program trace (BTM)	-	X	X	X
Port replacement	-	-	-	-
Data trace (write only)	-	-	X	X
Dynamic memory read/write	-	-	X	X
Data trace (read/write)	-	-	X	X
Data acquisition	-	-	X	X
Overrun Control	-	-	-	X
Memory substitution	-	-	-	-
Complex triggering	-	-	-	-
External memory substitution	-	-	-	-
Enable/disable trace based on a watchpoint	-	-	-	X

The JTAG and Nexus port will work both at 3.3 V and 5 V, according to the device power supply.

Boundary scan test is supported.

12.14.2 Debug port

The debug port is composed of a total number of 25 pads. For Nexus 3+ pin description, see Signal Description chapter.

The reset and ready pins that are often present as extensions to the JTAG port are not implemented.

12.15 Nexus Class 3+ Auxiliary Port

The Nexus Auxiliary Port provides real-time development class 3+ capabilities in compliance with the IEEE-ISTO 5001-2012 standard. This development support is supplied without requiring external address and data pins for internal visibility.

By default, after power-on reset, Nexus Auxiliary Port circuitry (controller) and the dedicated pad are disabled to avoid power consumption. It can only be enabled via a certain sequence given by the debugger. As soon as the Nexus Auxiliary Port is enabled, the Nexus pads are enabled.

NOTE

A full-port (16 MDOs) and reduced-port mode (12 MDOs) is supported and controlled via the Port Configuration Register (PCR) register.

Table 12-6. AUX Port Assignments

Instruction	Codes[5:0]	Instruction Summary
AUX_TAP_NPC_0	100001	Enables access to the NPC controller
AUX_TAP_NPC_1	100010	Enables access to the NPC controller used for low power handshake
AUX_TAP_MBIST	100100	Enables access to the MBIST TAP controller
AUX_TAP_JDC	100110	Enables access to the JDC TAP controller
AUX_TAP_HSM	100111	Enables access to the HSM TAP controller
AUX_TAP_Z4	101000	Enables access to the Z4 TAP controller
AUX_TAP_Z2	101010	Enables access to the Z2 TAP controller
AUX_TAP_NXMC	110100	Enables access to the NXMC TAP controller
AUX_TAP_PARALLEL	111100	Enables access to all the cores in daisy chain manner. For details, see Multi mode nexus TAP.

12.16 e200z2 Nexus 3

The e200z2 Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001-2012 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001-2012 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port in these processors

For details, see the e200z2 Nexus 3 chapter.

12.17 e200z4 Nexus 3

The e200z4 modules provide real-time development capabilities for e200z4d processors in compliance with the IEEE-ISTO 5001-2012 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

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

For details, see the Z4 Nexus Module chapter.

12.18 Nexus Crossbar Multi-master Client (NXMC)

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2011 standard. This module provides development support capabilities for SoCs without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard.

To provide Nexus data trace messaging from bus masters on the master ports of the crossbar (XBAR) that do not inherently have a trace client, a Nexus Crossbar Multi-Master Client (NXMC) monitors traffic into the master port of the XBAR. Trace messages transmitted over the Nexus interface (or stored in trace memory) include which master generated the message as well as an identifier for the pre-concentrator source of the message.

- Data Trace Message Formats supported: Data write/ read/ write sync/ read sync/ error messages.
- HSM master accesses filtering feature
- Watchpoint trigger input based messaging
- Watchpoint trigger out
- EVTO EVTI initiates sync message (message with complete address)
- All nexus trace control features just like any other client.
- No peripheral interface supported.
- It is an acceptable restriction that the user can not dynamically switch the input mux during run-time
- Watchpoints are to be implemented to permit enable/disable of trace

- The pin list specifies an EVTO pin for watch point signalling, which can be shared among all core nexus clients and the NXMC
- EVTO from the NXMC should be used for driving cross trigger, just like EVTO from other cores. NXMC is included in the cross triggering scheme for example, a watchpoint on a specified data access can trigger system (all other cores) to stop

For details, see the NXMC chapter.

Chapter 13
Reset and Boot Overview

13.1 Introduction

The chapter describes the global and local reset behavior of the device.

13.1.1 Global Reset

The global reset behavior refers to device reset behavior controlled by the MC_RGM which affects several modules of the device at the same time. The global reset is linked to the reset process described in Global Reset Process.

13.1.2 Local Reset

The local reset behavior refers to module reset behavior controlled by the MC_RGM or dedicated modules which affects only one module at a time. Local resets are described in the modules chapter.

13.1.3 Reset sources

As long as critical supply voltages are not reliable, the PMC provides the below mentioned signals to put the whole device into a POWERUP state. For more information, refer section, Reset Process Sequences

- LVD_FLASH
- LVD_IO_A_LO
- LVD_LV_PD0_hot
- LVD_LV_PD1_hot
- LVD_LV_PD2_hot

- POR_HV
- POR_LV

NOTE

On any of the above events, MC_RGM_DES[F_POR] will be set. Refer to PMCDIG_SGSR to detect the exact source of POWERUP for RGM. PMCDIG_SGSR[POR] indicates the POWERUP condition for the voltage regulator and MC_RGM_DES[F_POR] indicates the POWERUP for digital logic of the chip.

The following table lists the destructive and functional reset sources of the device.

Table 13-1. Destructive reset sources

Reset Source Mapping	Destructive Reset Source	Status	Reactions Possible	Bidirectional External reset generation/ Software control	Destructive Reset escalation enabled
1	Reserved				
2	HSM destructive reset	OK	Destructive	Yes/Read_only	Yes
3	Software generated destructive reset	OK	Destructive	Yes/Read_only	Yes
4	SSCM Secure reset request	OK	Destructive	Yes/Read_only	Yes
5	func_reset_escalation	OK	Destructive	Yes/Read_only	Yes
6	STCU unrecoverable fault	OK	Destructive	Yes/Read_only	Yes
7	SWT0 (flash initialization error)	OK	Destructive	Yes/Read_only	Yes
8	SWT1 reset request	OK	Destructive	Yes/Read_only	Yes

NOTE

Destructive reset escalation due to selective destructive reset sources can be disabled by configuring GPR_CTL[DEST_ESC_CTRL2], GPR_CTL[DEST_ESC_CTRL1], and GPR_CTL[DEST_ESC_CTRL0]. For more details see, GPR Chapter.

NOTE

STCU unrecoverable fault as source of destructive reset can be disabled from UTEST_MISC[SUF_DIS].

NOTE

No functional reset source can be demoted to Safe mode or interrupt in the LPU mode.

Table 13-2. Functional Reset Sources

Reset Source Mapping	Functional Reset Sources	Status	Reactions Possible	Bidirectional External reset generation/ Software control	Functional Reset escalation enabled
1	Reserved				
2	HSM generated functional reset	OK	Short Functional	No/Read_Write	Yes
3	Software generated functional reset	OK	Long/Short/ Functional	No/Read_Write	Yes
4	NMI from WKPU	OK	Long/Short Functional/Safe Request	No/Read_Write	Yes
5	JTAG_RST	OK	Long/Short Functional/Safe Request/Interrupt	No/Read_Write	Yes
6	Reset generated after self test is complete	OK	Long Functional	Yes/Read_only	Yes
7	FXOSC frequency out of range from CMU [1]	OK	Long/Short Functional/Safe Request/Interrupt	No/Read_Write	Yes
8	Long functional reset source by fault correction unit	OK	Long Functional	No/Read_Write	Yes
9	Short functional reset source by fault correction unit	OK	Short Functional	No/Read_Write	Yes
10	Z4 DBRST	OK	Long/Short Functional/Safe Request/Interrupt	No/Read_Write	Yes
11	Reserved				
12	Z2 DBRST	OK	Long/Short Functional/Safe Request/Interrupt	No/Read_Write	Yes
13	LVD_IO_A_HI	OK	Long/Short Functional/Safe Request/Interrupt	No/Read_Write	Yes
14	HVD_LV_cold	OK	Long/Short Functional/Safe Request/Interrupt	No/Read_Write	Yes
15	Reserved				
16	Reserved				
17	LVD_LV_PD2_cold	OK	Long/Short Functional/Safe Request/Interrupt	No/Read_Write	Yes

1. Reaction for this reset source should never be configured as interrupt or safe mode.

NOTE

HVD_LV_cold, LVD_LV_PD2_cold, and LVD_IO_A_HI as source of functional reset can be disabled from PMC_MISC[REE] bits. Also HVD feature can be disabled/ enabled from UTEST_MISC[HVD_EN]

During power-up, all functional terminal are maintained into a known state as described within Table 13-3

Table 13-3. Functional terminal state during power-up and reset

TERMINAL TYPE[1]	POWERUP pad state[2]	RESET pad state	DEFAULT pad state[3]	Comments
RESET	strong pull-down	strong pull-down	weak pull-up	functional reset pad.
PORST[4]	Weak pull down	Weak pull up	weak pull-up	power on reset pad.
GPIO	high impedance	high impedance	high impedance	by default, but configurable for STANDBY exit
ANALOG	high impedance	high impedance	high impedance	-
EOUT0, EOUT1	high impedance	high impedance	high impedance	
TCK	high impedance	weak pull-up	weak pull-up	-
TMS	high impedance	weak pull-up	weak pull-up	-
TDI	high impedance	weak pull-up	weak pull-up	-
TDO	high impedance	high impedance	high impedance	-
TCK_ALT	high impedance	weak pull-up	weak pull-up	-
TMS_ALT	high impedance	weak pull-up	weak pull-up	-
TDI_ALT	high impedance	weak pull-up	weak pull-up	-
TDO_ALT	high impedance	high impedance	high impedance	-

1. refer to pinout information for terminal type
2. POWERUP state is maintained until supply cross the power-on reset threshold: POR_LV for LV supply, POR_HV for high voltage supply.
3. before software configuration
4. PORST is not driven internally, so the state needs to be maintained externally on board

13.2 Global Reset Process

This section describes the global reset process.

13.2.1 Overview

The global reset process is a sequence of several reset phases. Each reset phase has a specific entry condition, a specific exit condition, and a specific device reset behavior, which is different among the reset phases. The reset phases are executed in a specific

order, thus building the actual global reset process. There are several modules contributing to the global reset process. All modules of the device can be affected by the global reset process.

13.2.2 Reset Process Modules

All modules which can enter, gate, or control the reset process are listed in Table 13-4 together with their type of contribution. The types of contribution are described in Table 13-5.

Table 13-4. Reset Process Modules

Module	Contribution Type		
	event	gating	control
Reset Generation Module (MC_RGM)	Yes	Yes	Yes
Power Management Digital Controller (PMC-Dig)	Yes	Yes	Yes
Mode Entry Module (MC_ME)	Yes	—	—
System Status and Configuration Module (SSCM)	—	Yes	—
Self-Test Control Unit	Yes	—	—
Flash Memory	—	Yes	—
MC_CGM	—	Yes	—
Software Watchdog Timeout (SWT0)	Yes	—	—

Table 13-5. Module Contribution Type

Contribution Type	Description
event	The module may generate events which may trigger the entry into a reset state, if the event configuration in the MC_RGM allows this.
gating	The module can prevent the exit of a reset state if it has not reached a certain internal state.
control	The module controls the sequence of the reset process.

13.2.2.1 Reset Generation Module (MC_RGM)

The MC_RGM is one of the two controlling module for the reset process. The MC_RGM can receive events from other modules and starts the reset process based on the configuration for these events. The event configuration defines the reset phase at which start the reset process starts. The MC_RGM controls the reset process sequence based on the gating information it receives from the gating modules.

13.2.2.2 Power Management Controller (PMC)

The PMC provides different voltage detect signals for the reset process control.

As long as the supply voltages are not reliable, the PMC provides following signals to allow the device to enter the various phases of reset process. For details, refer Reset sources

13.2.2.3 Mode Entry Module (MC_ME)

The MC_ME can generate events to enter the reset process. These events can be triggered only by application software.

13.2.2.4 System Status and Configuration Module (SSCM)

The SSCM is started automatically during a specific phase of the reset process. The SSCM fetches DCF records from the flash memory and distributes the read values via an internal DCF bus to the related modules. As long as the SSCM is still fetching DCF records, this specific reset phases can not be exited.

13.2.2.5 Self-Test Control Unit (STCU)

The STCU controls the execution of the start-up and shut-down self-test. Depending on its configuration the STCU is started when PHASE3[DEST] is left or when software triggers the self-test. The STCU generates a functional event when the self-testing has been completed.

STCU can also generate a STCU unrecoverable fault (SUF). If this fault occurs during start-up self test, it is intended to cause the start-up self test to be executed again. This is done via a ‘destructive’ reset request , if it is not disabled through UTEST_MISC[SUF_DIS]. If this fault occurs repeatedly, the ‘destructive’ reset escalator

in the MC_RGM will increment until it finally keeps the chip in reset until the next power-on reset. This is to prevent a start-up self-test pass that occurs after numerous failures to allow an unreliable chip to start execution and thus become a safety risk.

If this fault occurs during on-line or shut-down self-test , the handling is done by the FCCU. In this case, no ‘destructive’ reset request is generated.

13.2.2.6 Fault Collection and Control Unit (FCCU)

The FCCU generates reset events. Refer to Fault inputs for sources of fault. These can be configured through FCCU register interface for generating reset event.

13.2.2.7 MC_CGM

PHASE3 RESET will not exit if all system dividers are not aligned. This is internally checked by MC_CGM module.

13.2.2.8 Software Watchdog Timeout (SWT0)

SWT0 becomes active during PHASE2[FUNC]/PHASE2[DEST] RESET to monitor flash initialization failure. During normal operation, the flash indicates that it has completed its initialization process and exits PHASE2[FUNC]/PHASE2[DEST] RESET in reset sequence process.

In order to ensure that this happens within a reasonable time, the SWT is brought out of reset for PHASE2[FUNC]/PHASE2[DEST] RESET. So, if the SWT times out during this time, the flash’s initialization sequence was not able to complete. This is also referred as SWT0 (flash initialization error).

In order to trigger another attempt at initializing the flash, a ‘destructive’ reset is generated. If it occurs repeatedly, the ‘destructive’ reset escalator in the MC_RGM will increment until it finally keeps the chip in reset until the next power-on reset. This is to prevent unnecessary power consumption due to reset cycling.

13.2.3 Reset Process Sequences

This section specifies the reset processes sequences. An overview of the reset process sequences is given in the following figure. When the reset process has reached the IDLE state, the device is operational.

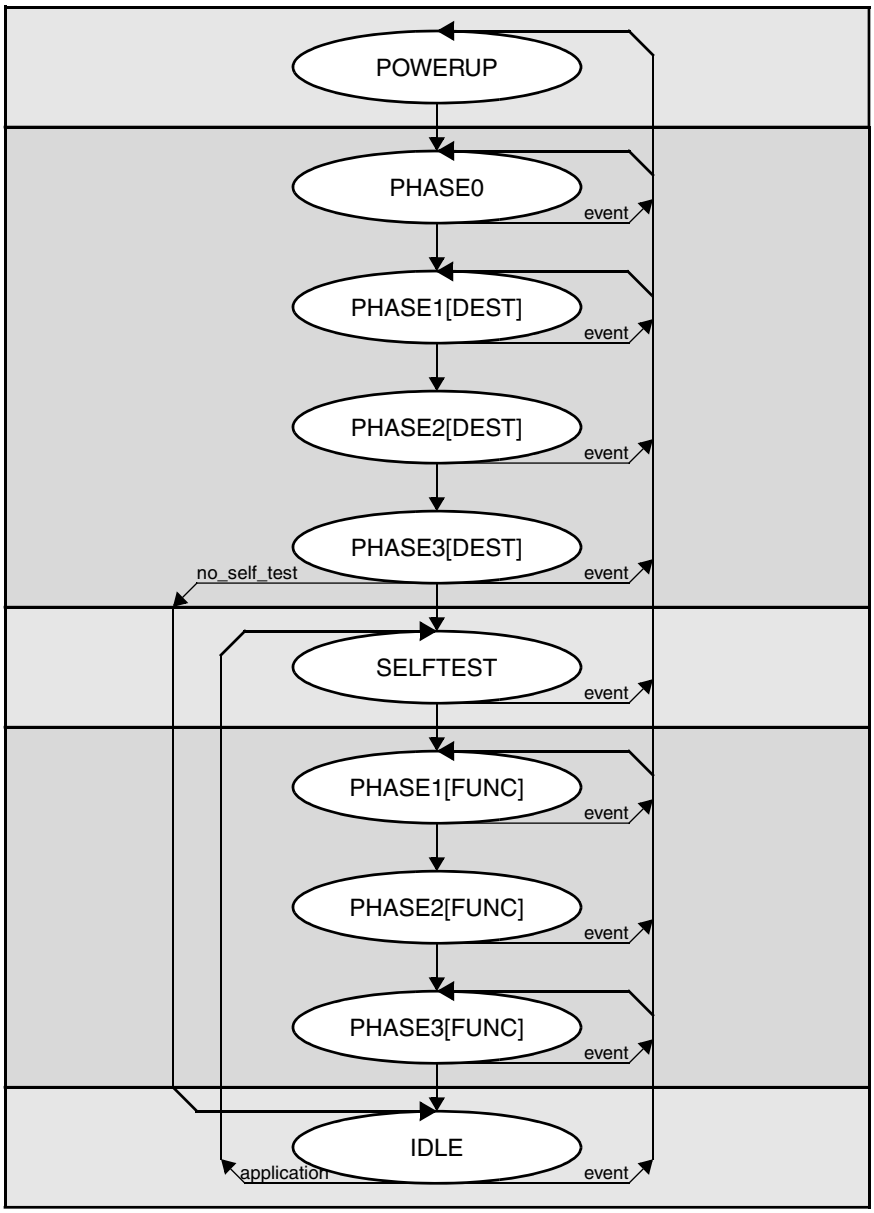


Figure 13-1. Reset Process Sequences

13.2.4 Reset Process State Transitions

This sections specifies the transitions between the reset states in the reset process. For each state, the state entry, the state execution, and the state exit is described.

13.2.4.1 POWERUP

13.2.4.1.1 POWERUP State Entry

This state is entered from any state if at least one of the following events appear

- Power is applied to the unpowered device.
- PMC has detected a low power condition that is lower than the minimum operating voltage

13.2.4.1.2 POWERUP State Execution

During this state, the PMC is waiting for stable power supply.

PMC will check that all logic domains are above min threshold voltage, referred as LVD, and then release power up for MC_RGM

Output pins are undriven and inputs are at default value. For details on state of each pad on reset, refer the device I/O Signal Description and Input Multiplexing tables attached to this document as an Excel file.

13.2.4.1.3 POWERUP State Duration

There are no bounds on the duration of this state.

13.2.4.1.4 POWERUP State Exit

Table 13-6. POWERUP Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	(1) The PORST pin is high in internal regulation mode and transitions from low to high when external supply is stable. Refer to the device datasheet for specifications. and (2) the PMC module indicates all supplies are in the operational range	FIRC MC_RGM PMC-Dig MC_CGM

NOTE

It is recommended to drive PORST pin when using external regulation mode, such that when external VDD_LV is not stable, PORST is driven LOW externally, and once VDD_LV is stable, PORST is driven HIGH. Alternatively, another equally robust solution is to connect PORST pin to VDD_HV_A supply. In this latter case, multiple voltage monitor resets may be seen during power-up/power-down in presence of a non-monotonic VDD_LV supply.

13.2.4.2 PHASE0

13.2.4.2.1 PHASE0 State Entry

This state can be entered whenever a destructive reset source is active. See Reset Sources section of the Reset Generation Module (MC_RGM) chapter.

13.2.4.2.2 PHASE0 State Execution

All trimming bits are reset to their default values.

13.2.4.2.3 PHASE0 State Duration

The maximum state of this duration depends on when the gates of this phase are released. Following are the gates which are required to be released

- FIRC stabilization counter should expire. Maximum count is 16 cycles of FIRC untrimmed clock

13.2.4.2.4 PHASE0 State Exit

Table 13-7. PHASE0 Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE1[DEST]	(1) The MC_RGM indicates that all sources of enabled destructive reset events are deasserted, and (2) the FIRC module indicates the generation of a stable clock,	STCU

13.2.4.3 PHASE1[DEST]

13.2.4.3.1 PHASE1[DEST] State Entry

This state can be entered from state

- PHASE0,
- PHASE1[FUNC],
- PHASE2[FUNC],

- PHASE3[FUNC],
- IDLE.

13.2.4.3.2 PHASE1[DEST] State Execution

There are no state changes in any module, only a counter in the MC_RGM is running.

13.2.4.3.3 PHASE1[DEST] State Duration

The duration of this state is 8 FIRC clock cycles.

13.2.4.3.4 PHASE1[DEST] State Exit

Table 13-8. PHASE1[DEST] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE2[DEST]	8 FIRC clocks passed by after state entry.	FLASH - starts initialization sequence FCCU - enters configuration state

13.2.4.4 PHASE2[DEST]

13.2.4.4.1 PHASE2[DEST] State Entry

This state can be entered from state

- PHASE1[DEST].

13.2.4.4.2 PHASE2[DEST] State Execution

FLASH executes initialization and confirms that it is ready for access. This information is used to gate phase2.

13.2.4.4.3 PHASE2[DEST] State Duration

There is no counter in MC_RGM, it is completely gate dependent.

13.2.4.4.4 PHASE2[DEST] State Exit

Table 13-9. PHASE2[DEST] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE3[DEST]	The FLASH module indicates that it is ready for access.	SSCM - starts device configuration and boot location search

13.2.4.5 PHASE3[DEST]

13.2.4.5.1 PHASE3[DEST] State Entry

This state can be entered only from state

- PHASE2[DEST].

13.2.4.5.2 PHASE3[DEST] State Execution

SSCM runs device configuration and boot location search.

The PMC adjusts its LVD and HVD levels according to the DCF record content after SSCM confirms that the DCF records are read from Flash.

Gates of phase3

- SSCM confirm DCF records are read from flash.
- HSM's RAM initialization is done.
- MC_CGM confirms that all system clock dividers are aligned.
- Flash is ready for access. This is again a gate for phase3 to ensure that flash is available when a short functional reset occurs.

13.2.4.5.3 PHASE3[DEST] State Duration

Maximum duration of this phase is determined by the gates. Its duration can be extended if external reset pin is kept asserted. Minimum phase duration is 40 FIRC clock cycles.

13.2.4.5.4 PHASE3[DEST] State Exit

Table 13-10. PHASE3[DEST] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
SELFTEST	(1) The FLASH module indicates availability of full speed array access, and (2) the SSCM module indicates device configuration is done, and (3) the HSM module indicates that HSM RAM initialization is done, and (4) the CGM module indicates that all system clock dividers are aligned, and (5) the STCU module is configured to execute a start-up self test.	BIST on - FCCU - CORES - PERIPHERALS
IDLE	(1) The FLASH module indicates availability of array access, and (2) the SSCM module indicates device configuration is done, and (3) the HSM module indicates that HSM RAM initialization is done, and (4) the CGM module indicates that all system clock dividers are aligned.	Z4A CORE ALL OTHER PERIPHERALS

13.2.4.6 SELFTEST

13.2.4.6.1 SELFTEST State Entry

This state can be entered only from state

- PHASE3[DEST],
- IDLE.

13.2.4.6.2 SELFTEST State Execution

The start-up self test is executed as configured in the STCU by the DCF record content.

13.2.4.6.3 SELFTEST State Duration

The SELFTEST runtimes with respect to PLL and IRC can be calculated as below:

SELFTEST state duration = reset phases + SELFTEST RUN (mbist) + reset phases

The minimum SELFTEST state duration (@PLL 40 Mhz)= 6.4795ms (0.194ms + 6.1047ms + 0.182ms), where

mbist all memories parallel= 4.299ms

Maximum selftest state duration (@IRC 16Mhz)= 14.30279ms (0.194ms + 13.9323ms + 0.182ms), where

mbist all memories parallel= 10.748ms

13.2.4.6.4 SELFTEST State Exit

Table 13-11. SELFTEST Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[FUNC]	STCU module indicates end of self test.	

13.2.4.7 PHASE1[FUNC]

13.2.4.7.1 PHASE1[FUNC] State Entry

This state can be entered from states

- SELFTEST,
- PHASE3[FUNC],
- IDLE.

13.2.4.7.2 PHASE1[FUNC] State Execution

There are no state changes in any module, only a counter in the MC_RGM is running.

13.2.4.7.3 PHASE1[FUNC] State Duration

The duration of this state is 8 FIRC clock cycles.

13.2.4.7.4 PHASE1[FUNC] State Exit

Table 13-12. PHASE1[FUNC] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[DEST]	The MC_RGM has detected that the source of the enabled long functional external reset event is asserted.	
PHASE2[FUNC]	(1) The MC_RGM indicates that all sources of enabled long functional reset events are deasserted, and (2) 8 FIRC clocks passed by after state entry.	FLASH - starts initialization sequence FCCU -enters configuration state

13.2.4.8 PHASE2[FUNC]

13.2.4.8.1 PHASE2[FUNC] State Entry

This state can be entered from state

- PHASE1[FUNC].

13.2.4.8.2 PHASE2[FUNC] State Execution

Flash executes initialization for array access.

13.2.4.8.3 PHASE2[FUNC] State Duration

There is no counter in MC_RGM, it is completely gate dependent.

13.2.4.8.4 PHASE2[FUNC] State Exit

Table 13-13. PHASE2[FUNC] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[DEST]	The MC_RGM has detected that the source of the enabled long functional external reset event is asserted.	
PHASE1[FUNC]	The MC_RGM has detected that the sources of an enabled long functional reset event is asserted.	
PHASE3[FUNC]	The FLASH module indicates that it is ready for access.	SSCM - starts device configuration

13.2.4.9 PHASE3[FUNC]

13.2.4.9.1 PHASE3[FUNC] State Entry

This state can be entered from states

- PHASE2[FUNC]
- IDLE

13.2.4.9.2 PHASE3[FUNC] State Execution

SSCM runs device configuration, if state was entered from PHASE2[FUNC].

The FCCU executes its configuration sequence if PHASE3[FUNC] was entered via the SELFTEST state.

The FLASH indicates that it is ready for access.

13.2.4.9.3 PHASE3[FUNC] State Duration

Duration of this phase is determined by the gates. Its duration can be extended if external reset pin is kept asserted.

13.2.4.9.4 PHASE3[FUNC] State Exit

Table 13-14. PHASE3[FUNC] Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[DEST]	The MC_RGM has detected that the source of the enabled long functional external reset event is asserted.	
PHASE1[FUNC]	The MC_RGM has detected that the sources of an enabled long functional reset event is asserted.	
IDLE	(1) The FLASH module indicates availability of array access, and (2) the SSCM module indicates device configuration is done, and (3) the HSM module indicates that HSM RAM initialization is done and (4) the CGM module indicates that all system clock dividers are aligned.	Z4A CORE ALL OTHER PERIPHERALS

13.2.4.10 IDLE

13.2.4.10.1 IDLE State Entry

The IDLE state can be entered only from states

- PHASE3[DEST]
- PHASE3[FUNC]

13.2.4.10.2 IDLE State Execution

During the IDLE state, the device is fully operational.

13.2.4.10.3 IDLE State Duration

There are no bounds on the duration of this state.

13.2.4.10.4 IDLE State Exit

The IDLE state next state table is shown in Table 13-15 the following table.

Table 13-15. IDLE Next State Table

Next State	Condition	Process/Module Started at Exit
PHASE0	The MC_RGM has detected that the source of an enabled destructive reset event is asserted.	
PHASE1[DEST]	The MC_RGM has detected that the source of the enabled long functional external reset event is asserted.	
PHASE1[FUNC]	The MC_RGM has detected that the source of an enabled long functional reset event is asserted.	
PHASE3[FUNC]	The MC_RGM has detected that the source of an enabled short functional reset event is asserted.	
SELFTEST	The application has triggered the start of the shut-down selftest.	

13.2.5 Module Status During Reset Process

This section specifies in Table 13-16 the status of the device modules during the reset process. The status abbreviation explained in Table 13-17.

Table 13-16. Module Status during reset states

	Global Reset State									
Module	POWER UP	PHASE0	PHASE 1[DEST]	PHASE 2[DEST]	PHASE 3[DEST]	SELFTEST	PHASE 1[FUNC]	PHASE 2[FUNC]	PHASE 3[FUNC]	IDLE
PMC	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON
FIRC	RST	ON	ON	ON	ON	ON	ON	ON	ON	ON
SIRC	RST	RST	ON	ON	ON	ON	ON	ON	ON	ON
MC_RGM	RST	ON	ON	ON	ON	ON	ON	ON	ON	ON
FXOSC	RST	RST	RST	RST	ON	ON	ON	ON	ON	ON
FLASH	RST	RST	RST	ON	ON	ON	RST	ON	ON	ON
FCCU	RST	RST	RST	ON	ON	BIST	RST	ON	ON	ON
RTC	RST	ON	ON	ON	ON	ON	ON	ON	ON	ON
STCU	RST	RST	RST	ON	ON	ON	ON	ON	ON	ON
Z4A CORE	RST	RST	RST	RST	RST	BIST	RST	RST	RST	ON
OTHERS	RST	RST	RST	RST	RST	BIST	RST	RST	RST	ON
SWT0	RST	RST	RST	ON	RST	ON	RST	ON	RST	ON

Table 13-17. Module Status Glossary

Module Status	Status Description
RST	Module is held in reset by the global reset process
BIST	Module is being tested or held in a non-functional state during self test execution as controlled by the STCU.
ON	Module is functional.

For information on impact of different phases of reset on individual modules, See *MPC5746C_Module_Reset_Status* spreadsheet attached to this document.

13.3 Boot

This following table lists the various conditions for booting the device.

Table 13-18. Booting conditions

Boot Condition	Booting Core	Boot vector
Power Up	Only Z4a	BAF code entry point - 0040_4100h
Exit from STANDBY/LPU_STANDBY to DRUN	Any core based on configuration of MC_ME_CCTLn	MC_ME_CADDRn
Exit from LPU STANDBY to LPU_RUN	Only Z2	SRAM_0 base address - 0x40000000
Entry from RUNx/DRUN to LPU_RUN	Only Z2	SRAM_0 base address - 0x40000000

13.3.1 Power Up

Refer BAF for details.

13.3.2 Exit from STANDBY/LPU_STANDBY to DRUN

MC_ME_CADDRn should be configured prior to entry into STANDBY or LPU modes.

13.3.2.1 Boot from RAM

SRAM location to be loaded in MC_ME_CADDRn. Standby exit to DRUN is faster if Flash is kept power down by configuring MC_ME_DRUN_MC[FLAON] to 2'b01 prior to entry into STANDBY or LPU modes.

13.3.2.2 Boot with BAF

Device can boot from BAF location, if the same is configured in MC_ME_CADDRn. Flash should be powered ON in DRUN if booting vector resides in flash address range.

NOTE

While configuring the device to boot from BAF after standby exit, it needs to be taken care of that UTEST_MISC[BAF_DIS] is not set.

13.3.3 Exit from LPU STANDBY to LPU_RUN

Refer Boot section.

13.3.4 Entry from RUNx/DRUN to LPU_RUN

Refer Boot section.

13.3.5 Interrupt Vector Prefix retention during STANDBY

In the Cores, the Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. However to use this IVPR feature on exit from STANDBY modes, the prefix value can be programmed in GPR_CTL1[IVPR] before STANDBY entry.

This register defines the base location of the interrupt vector table to the two cores z4 and z2.

When exiting Standby mode with boot vector pointing to RAM location, IVPR of the cores can be directed to RAM address location.

Chapter 14
Functional Safety

14.1 Introduction

The MPC5746C is developed according to ISO 26262 and has an integrated safety concept targeting safety-related systems requiring high safety integrity levels. In order to support the integration of the MPC5746C into safety-related systems, the following documentation is available:

- Reference Manual (MPC5746CRM) - Describes the MPC5746C functionality
- Data Sheet (MPC5746CDS) - Describes the operating conditions
- Safety Manual (MPC5746CSM) - Describes the MPC5746C safety concept and possible safety mechanisms (integrated in MPC5746C, system level hardware or system level software), as well as measures to reduce dependent failures.
- Dynamic FMEDA - Inductive analysis enabling customization of system level safety mechanisms, including the resulting safety metrics for ISO 26262 (SPFM, LFM and PMHF) and IEC 61508 (SFF and Beta IC Factor)
- FMEDA Report - Describes the FMEDA methodology and safety mechanisms supported in the FMEDA, including source of failure rates, failure modes and assumptions made during the analysis.

The Dynamic FMEDA and FMEDA report are available upon request. The MPC5746C is a SafeAssure solution; for further information regarding functional safety, visit www.nxp.com/safeassure.

Chapter 15
System Integration Unit Lite2 (SIUL2)

15.1 Introduction

15.1.1 Overview

The SIUL2 provides control over all the electrical pin controls and up to 32 ports with 16 bits of bidirectional, general-purpose input and output signals. One of the most important functions of the SIUL2 is that it enables the user to select the functions and electrical characteristics that appear on external device pins. It also controls the multiplexing of internal signals from one module to another and controls device I/O. It supports up to 32 external interrupts with trigger event configuration.

The module provides dedicated pad control to general-purpose pads that can be configured as either inputs or outputs. The SIUL2 module provides registers that enable user software to read values from GPIO pads configured as inputs, and write values to GPIO pads configured as outputs

- When configured as output, you can write to an internal register to control the state driven on the associated output pad.
- When configured as input, you can detect the state of the associated pad by reading the value from an internal register.
- When configured as input and output, the pad value can be read back, which can be used as a method of checking if the written value appeared on the pad.

To assist software development, GPIO data registers can be accessed using various mechanisms. These differing mechanisms allow support for port access or for bit manipulation without the need to use read-modify-write operations or access with Power Architecture reservation.

- Access to two 16 bit ports in one access

- Read/write access to a single bit
- A 16-bit Port write with a bit mask, using single 32-bit access.

NOTE

For any unavailable MSCR, transfer error for peripheral access will not be generated. However, 8- or 16- bit access to any valid MSCR will result in a transfer error.

The external interrupt sources can be configured at the MCU level to be used with any chip pad. From 0 to 31 of the interrupt sources can be configured to have a digital filter to reject short glitches on the inputs. The external interrupt requests map to the REQ pins in the device packages.

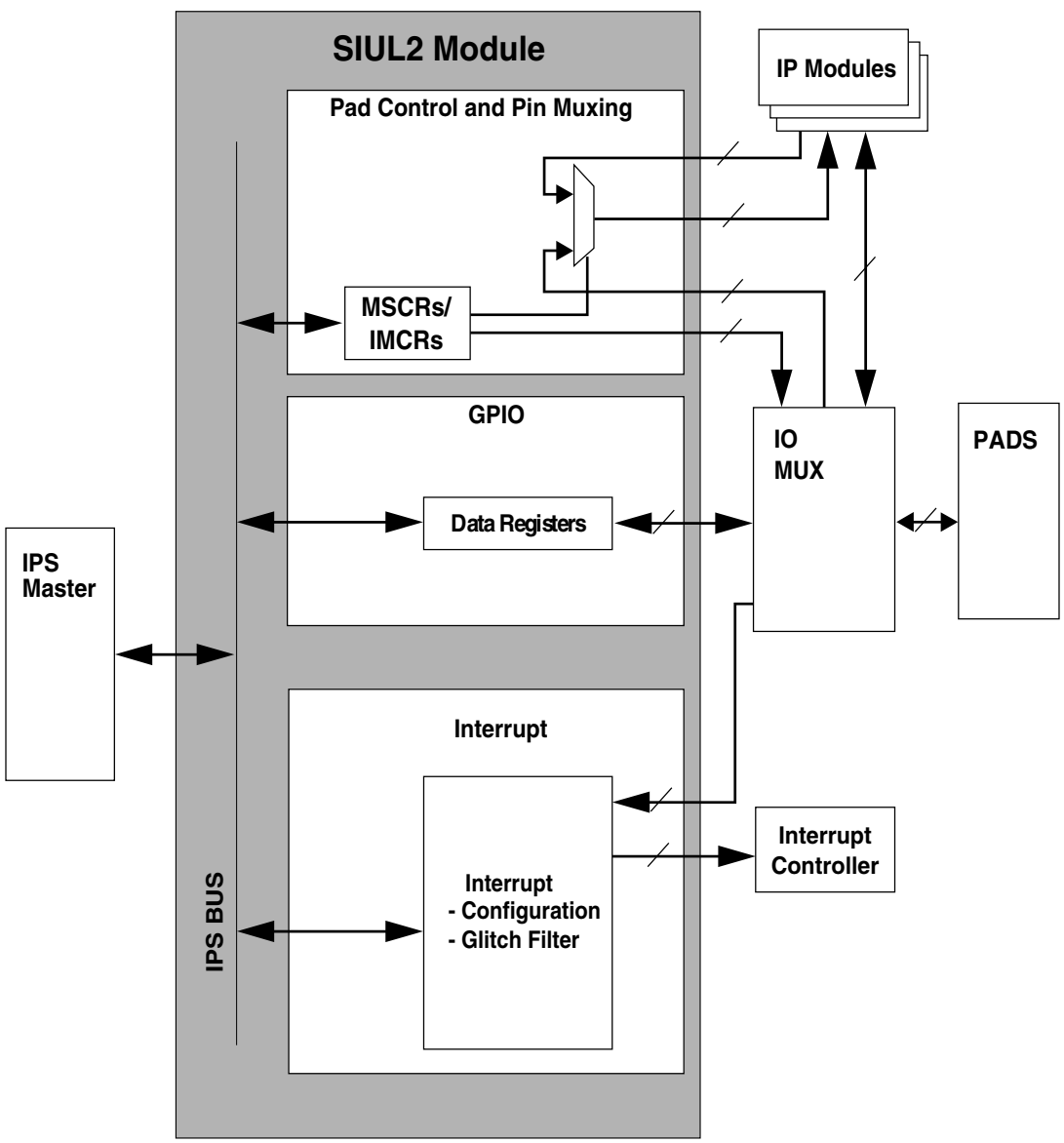


Figure 15-1. SIUL2 block diagram

15.1.2 Features

The SIUL2 supports these distinctive features:

- 1 to 32 GPIO ports with data control
  - Drive data to up to 16 independent I/O channels
  - Sample data from up to 16 independent I/O channels

Two 16-bit registers can be read/written with one access for a 32-bit port, if needed.

External interrupt request support with:

- 0 to 3 system interrupt vectors for 0 to 31 interrupt sources with independent interrupt masks. For 32 external interrupt sources (REQ pins), four groups have eight interrupt sources each, and each of the four groups is assigned one system interrupt vector.
- 0 to 31 programmable digital glitch filters, one for each REQ pin
- Edge detection

Additionally the SIUL2 contains the Multiplexed Signal Configuration Registers that configure the electrical parameters and settings of up to 264 functional pads. The number of these registers that is actually implemented varies by device. These registers configure the following pad features:

- Drive strength
- Open drain/source output enable
- Slew rate control
- Hysteresis control
- Internal pull control and pull selection
- Pin function assignment
- Control of analog path switches
- Safe mode behavior configuration

15.1.3 Register protection

The System Integration Unit Lite2 uses the Register Protection Scheme to protect the individual registers from accidental writes. The following registers can be protected:

- SIUL2 Interrupt Request Enable Register 0 (SIUL2_IRER0)
- SIUL2 Interrupt Request Select Register 0 (SIUL2_IRSR0)
- SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)
- SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)
- SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)
- SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0 to SIUL2_IFMCR31)
- SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)
- SIUL2 Multiplexed Single Configuration Register (SIUL2_MSCR0 to SIUL2_MSCR263 and SIUL2_IMCR0 to SIUL2_IMCR511): Only the specific MSCRs and IMCRs that are actually implemented on a device are protected.
- SIUL2 GPIO Pad Data Output Register (SIUL2_GPDOn)
- SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDOn)
- SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDOn)

A compiled list identifies the protected registers at specific addresses for this device. Please refer the Register protection configuration.

15.2 Memory map and register description

This section provides a detailed description of all registers accessible in the SIUL module in address order. Each description includes a standard register diagram. Details of register bit and field function follow the register diagrams, in bit order. MIDRn, IFMCRn, MSCRn, IFCP, MPGPDOn(writes) are 32-bit accessible only registers.

The following table gives an overview on the SIUL2 registers implemented.

NOTE

Undocumented register spaces in the SIUL2 memory map, including addresses shown as "blank," are reserved. Reserved registers read as 0, and writes to these registers have no effect. A transfer error is issued for attempts to access a completely reserved register space. A write to reserved register bit has no effect.

SIUL2 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	SIUL2 MCU ID Register #1 (SIUL2_MIDR1)	32	R	See section	15.2.1/338
8	SIUL2 MCU ID Register #2 (SIUL2_MIDR2)	32	R	3A00_4300h	15.2.2/339
10	SIUL2 Interrupt Status Flag Register0 (SIUL2_ISR0)	32	w1c	0000_0000h	15.2.3/340
18	SIUL2 Interrupt Request Enable Register0 (SIUL2_IRER0)	32	R/W	0000_0000h	15.2.4/344
20	SIUL2 Interrupt Request Select Register0 (SIUL2_IRSR0)	32	R/W	0000_0000h	15.2.5/347
28	SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)	32	R/W	0000_0000h	15.2.6/350
30	SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)	32	R/W	0000_0000h	15.2.7/353
38	SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)	32	R/W	0000_0000h	15.2.8/356
40	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR0)	32	R/W	0000_0000h	15.2.9/358
44	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR1)	32	R/W	0000_0000h	15.2.9/358
48	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR2)	32	R/W	0000_0000h	15.2.9/358
4C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR3)	32	R/W	0000_0000h	15.2.9/358
50	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR4)	32	R/W	0000_0000h	15.2.9/358

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
54	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR5)	32	R/W	0000_0000h	15.2.9/358
58	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR6)	32	R/W	0000_0000h	15.2.9/358
5C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR7)	32	R/W	0000_0000h	15.2.9/358
60	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR8)	32	R/W	0000_0000h	15.2.9/358
64	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR9)	32	R/W	0000_0000h	15.2.9/358
68	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR10)	32	R/W	0000_0000h	15.2.9/358
6C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR11)	32	R/W	0000_0000h	15.2.9/358
70	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR12)	32	R/W	0000_0000h	15.2.9/358
74	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR13)	32	R/W	0000_0000h	15.2.9/358
78	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR14)	32	R/W	0000_0000h	15.2.9/358
7C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR15)	32	R/W	0000_0000h	15.2.9/358
80	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR16)	32	R/W	0000_0000h	15.2.9/358
84	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR17)	32	R/W	0000_0000h	15.2.9/358
88	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR18)	32	R/W	0000_0000h	15.2.9/358
8C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR19)	32	R/W	0000_0000h	15.2.9/358
90	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR20)	32	R/W	0000_0000h	15.2.9/358
94	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR21)	32	R/W	0000_0000h	15.2.9/358
98	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR22)	32	R/W	0000_0000h	15.2.9/358
9C	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR23)	32	R/W	0000_0000h	15.2.9/358
A0	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR24)	32	R/W	0000_0000h	15.2.9/358
A4	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR25)	32	R/W	0000_0000h	15.2.9/358
A8	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR26)	32	R/W	0000_0000h	15.2.9/358

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AC	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR27)	32	R/W	0000_0000h	15.2.9/358
B0	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR28)	32	R/W	0000_0000h	15.2.9/358
B4	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR29)	32	R/W	0000_0000h	15.2.9/358
B8	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR30)	32	R/W	0000_0000h	15.2.9/358
BC	SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR31)	32	R/W	0000_0000h	15.2.9/358
C0	SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)	32	R/W	0000_0000h	15.2.10/359
240	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR0)	32	R/W	See section	15.2.11/360
244	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR1)	32	R/W	See section	15.2.11/360
248	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR2)	32	R/W	See section	15.2.11/360
24C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR3)	32	R/W	See section	15.2.11/360
250	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR4)	32	R/W	See section	15.2.11/360
254	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR5)	32	R/W	See section	15.2.11/360
258	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR6)	32	R/W	See section	15.2.11/360
25C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR7)	32	R/W	See section	15.2.11/360
260	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR8)	32	R/W	See section	15.2.11/360
264	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR9)	32	R/W	See section	15.2.11/360
268	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR10)	32	R/W	See section	15.2.11/360
26C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR11)	32	R/W	See section	15.2.11/360
270	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR12)	32	R/W	See section	15.2.11/360
274	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR13)	32	R/W	See section	15.2.11/360
278	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR14)	32	R/W	See section	15.2.11/360
27C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR15)	32	R/W	See section	15.2.11/360

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
280	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR16)	32	R/W	See section	15.2.11/360
284	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR17)	32	R/W	See section	15.2.11/360
288	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR18)	32	R/W	See section	15.2.11/360
28C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR19)	32	R/W	See section	15.2.11/360
290	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR20)	32	R/W	See section	15.2.11/360
294	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR21)	32	R/W	See section	15.2.11/360
298	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR22)	32	R/W	See section	15.2.11/360
29C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR23)	32	R/W	See section	15.2.11/360
2A0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR24)	32	R/W	See section	15.2.11/360
2A4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR25)	32	R/W	See section	15.2.11/360
2A8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR26)	32	R/W	See section	15.2.11/360
2AC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR27)	32	R/W	See section	15.2.11/360
2B0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR28)	32	R/W	See section	15.2.11/360
2B4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR29)	32	R/W	See section	15.2.11/360
2B8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR30)	32	R/W	See section	15.2.11/360
2BC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR31)	32	R/W	See section	15.2.11/360
2C0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR32)	32	R/W	See section	15.2.11/360
2C4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR33)	32	R/W	See section	15.2.11/360
2C8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR34)	32	R/W	See section	15.2.11/360
2CC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR35)	32	R/W	See section	15.2.11/360
2D0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR36)	32	R/W	See section	15.2.11/360
2D4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR37)	32	R/W	See section	15.2.11/360

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2D8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR38)	32	R/W	See section	15.2.11/360
2DC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR39)	32	R/W	See section	15.2.11/360
2E0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR40)	32	R/W	See section	15.2.11/360
2E4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR41)	32	R/W	See section	15.2.11/360
2E8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR42)	32	R/W	See section	15.2.11/360
2EC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR43)	32	R/W	See section	15.2.11/360
2F0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR44)	32	R/W	See section	15.2.11/360
2F4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR45)	32	R/W	See section	15.2.11/360
2F8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR46)	32	R/W	See section	15.2.11/360
2FC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR47)	32	R/W	See section	15.2.11/360
300	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR48)	32	R/W	See section	15.2.11/360
304	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR49)	32	R/W	See section	15.2.11/360
308	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR50)	32	R/W	See section	15.2.11/360
30C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR51)	32	R/W	See section	15.2.11/360
310	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR52)	32	R/W	See section	15.2.11/360
314	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR53)	32	R/W	See section	15.2.11/360
318	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR54)	32	R/W	See section	15.2.11/360
31C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR55)	32	R/W	See section	15.2.11/360
320	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR56)	32	R/W	See section	15.2.11/360
324	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR57)	32	R/W	See section	15.2.11/360
328	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR58)	32	R/W	See section	15.2.11/360
32C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR59)	32	R/W	See section	15.2.11/360

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
330	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR60)	32	R/W	See section	15.2.11/360
334	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR61)	32	R/W	See section	15.2.11/360
338	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR62)	32	R/W	See section	15.2.11/360
33C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR63)	32	R/W	See section	15.2.11/360
340	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR64)	32	R/W	See section	15.2.11/360
344	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR65)	32	R/W	See section	15.2.11/360
348	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR66)	32	R/W	See section	15.2.11/360
34C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR67)	32	R/W	See section	15.2.11/360
350	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR68)	32	R/W	See section	15.2.11/360
354	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR69)	32	R/W	See section	15.2.11/360
358	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR70)	32	R/W	See section	15.2.11/360
35C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR71)	32	R/W	See section	15.2.11/360
360	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR72)	32	R/W	See section	15.2.11/360
364	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR73)	32	R/W	See section	15.2.11/360
368	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR74)	32	R/W	See section	15.2.11/360
36C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR75)	32	R/W	See section	15.2.11/360
370	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR76)	32	R/W	See section	15.2.11/360
374	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR77)	32	R/W	See section	15.2.11/360
378	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR78)	32	R/W	See section	15.2.11/360
37C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR79)	32	R/W	See section	15.2.11/360
380	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR80)	32	R/W	See section	15.2.11/360
384	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR81)	32	R/W	See section	15.2.11/360

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
388	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR82)	32	R/W	See section	15.2.11/360
38C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR83)	32	R/W	See section	15.2.11/360
390	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR84)	32	R/W	See section	15.2.11/360
394	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR85)	32	R/W	See section	15.2.11/360
398	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR86)	32	R/W	See section	15.2.11/360
39C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR87)	32	R/W	See section	15.2.11/360
3A0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR88)	32	R/W	See section	15.2.11/360
3A4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR89)	32	R/W	See section	15.2.11/360
3A8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR90)	32	R/W	See section	15.2.11/360
3AC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR91)	32	R/W	See section	15.2.11/360
3B0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR92)	32	R/W	See section	15.2.11/360
3B4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR93)	32	R/W	See section	15.2.11/360
3B8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR94)	32	R/W	See section	15.2.11/360
3BC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR95)	32	R/W	See section	15.2.11/360
3C0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR96)	32	R/W	See section	15.2.11/360
3C4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR97)	32	R/W	See section	15.2.11/360
3C8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR98)	32	R/W	See section	15.2.11/360
3CC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR99)	32	R/W	See section	15.2.11/360
3D0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR100)	32	R/W	See section	15.2.11/360
3D4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR101)	32	R/W	See section	15.2.11/360
3D8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR102)	32	R/W	See section	15.2.11/360
3DC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR103)	32	R/W	See section	15.2.11/360

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3E0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR104)	32	R/W	See section	15.2.11/360
3E4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR105)	32	R/W	See section	15.2.11/360
3E8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR106)	32	R/W	See section	15.2.11/360
3EC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR107)	32	R/W	See section	15.2.11/360
3F0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR108)	32	R/W	See section	15.2.11/360
3F4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR109)	32	R/W	See section	15.2.11/360
3F8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR110)	32	R/W	See section	15.2.11/360
3FC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR111)	32	R/W	See section	15.2.11/360
400	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR112)	32	R/W	See section	15.2.11/360
404	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR113)	32	R/W	See section	15.2.11/360
408	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR114)	32	R/W	See section	15.2.11/360
40C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR115)	32	R/W	See section	15.2.11/360
410	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR116)	32	R/W	See section	15.2.11/360
414	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR117)	32	R/W	See section	15.2.11/360
418	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR118)	32	R/W	See section	15.2.11/360
41C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR119)	32	R/W	See section	15.2.11/360
420	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR120)	32	R/W	See section	15.2.11/360
424	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR121)	32	R/W	See section	15.2.11/360
428	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR122)	32	R/W	See section	15.2.11/360
42C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR123)	32	R/W	See section	15.2.11/360
430	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR124)	32	R/W	See section	15.2.11/360
434	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR125)	32	R/W	See section	15.2.11/360

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
438	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR126)	32	R/W	See section	15.2.11/360
43C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR127)	32	R/W	See section	15.2.11/360
440	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR128)	32	R/W	See section	15.2.11/360
444	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR129)	32	R/W	See section	15.2.11/360
448	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR130)	32	R/W	See section	15.2.11/360
44C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR131)	32	R/W	See section	15.2.11/360
450	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR132)	32	R/W	See section	15.2.11/360
454	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR133)	32	R/W	See section	15.2.11/360
458	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR134)	32	R/W	See section	15.2.11/360
45C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR135)	32	R/W	See section	15.2.11/360
460	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR136)	32	R/W	See section	15.2.11/360
464	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR137)	32	R/W	See section	15.2.11/360
468	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR138)	32	R/W	See section	15.2.11/360
46C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR139)	32	R/W	See section	15.2.11/360
470	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR140)	32	R/W	See section	15.2.11/360
474	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR141)	32	R/W	See section	15.2.11/360
478	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR142)	32	R/W	See section	15.2.11/360
47C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR143)	32	R/W	See section	15.2.11/360
480	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR144)	32	R/W	See section	15.2.11/360
484	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR145)	32	R/W	See section	15.2.11/360
488	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR146)	32	R/W	See section	15.2.11/360
48C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR147)	32	R/W	See section	15.2.11/360

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
490	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR148)	32	R/W	See section	15.2.11/360
494	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR149)	32	R/W	See section	15.2.11/360
498	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR150)	32	R/W	See section	15.2.11/360
49C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR151)	32	R/W	See section	15.2.11/360
4A0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR152)	32	R/W	See section	15.2.11/360
4A4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR153)	32	R/W	See section	15.2.11/360
4A8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR154)	32	R/W	See section	15.2.11/360
4AC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR155)	32	R/W	See section	15.2.11/360
4B0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR156)	32	R/W	See section	15.2.11/360
4B4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR157)	32	R/W	See section	15.2.11/360
4B8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR158)	32	R/W	See section	15.2.11/360
4BC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR159)	32	R/W	See section	15.2.11/360
4C0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR160)	32	R/W	See section	15.2.11/360
4C4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR161)	32	R/W	See section	15.2.11/360
4C8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR162)	32	R/W	See section	15.2.11/360
4CC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR163)	32	R/W	See section	15.2.11/360
4D0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR164)	32	R/W	See section	15.2.11/360
4D4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR165)	32	R/W	See section	15.2.11/360
4D8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR166)	32	R/W	See section	15.2.11/360
4DC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR167)	32	R/W	See section	15.2.11/360
4E0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR168)	32	R/W	See section	15.2.11/360
4E4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR169)	32	R/W	See section	15.2.11/360

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4E8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR170)	32	R/W	See section	15.2.11/360
4EC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR171)	32	R/W	See section	15.2.11/360
4F0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR172)	32	R/W	See section	15.2.11/360
4F4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR173)	32	R/W	See section	15.2.11/360
4F8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR174)	32	R/W	See section	15.2.11/360
4FC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR175)	32	R/W	See section	15.2.11/360
500	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR176)	32	R/W	See section	15.2.11/360
504	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR177)	32	R/W	See section	15.2.11/360
508	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR178)	32	R/W	See section	15.2.11/360
50C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR179)	32	R/W	See section	15.2.11/360
510	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR180)	32	R/W	See section	15.2.11/360
514	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR181)	32	R/W	See section	15.2.11/360
518	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR182)	32	R/W	See section	15.2.11/360
51C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR183)	32	R/W	See section	15.2.11/360
520	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR184)	32	R/W	See section	15.2.11/360
524	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR185)	32	R/W	See section	15.2.11/360
528	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR186)	32	R/W	See section	15.2.11/360
52C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR187)	32	R/W	See section	15.2.11/360
530	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR188)	32	R/W	See section	15.2.11/360
534	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR189)	32	R/W	See section	15.2.11/360
538	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR190)	32	R/W	See section	15.2.11/360
53C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR191)	32	R/W	See section	15.2.11/360

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
540	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR192)	32	R/W	See section	15.2.11/360
544	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR193)	32	R/W	See section	15.2.11/360
548	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR194)	32	R/W	See section	15.2.11/360
54C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR195)	32	R/W	See section	15.2.11/360
550	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR196)	32	R/W	See section	15.2.11/360
554	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR197)	32	R/W	See section	15.2.11/360
558	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR198)	32	R/W	See section	15.2.11/360
55C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR199)	32	R/W	See section	15.2.11/360
560	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR200)	32	R/W	See section	15.2.11/360
564	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR201)	32	R/W	See section	15.2.11/360
568	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR202)	32	R/W	See section	15.2.11/360
56C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR203)	32	R/W	See section	15.2.11/360
570	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR204)	32	R/W	See section	15.2.11/360
574	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR205)	32	R/W	See section	15.2.11/360
578	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR206)	32	R/W	See section	15.2.11/360
57C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR207)	32	R/W	See section	15.2.11/360
580	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR208)	32	R/W	See section	15.2.11/360
584	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR209)	32	R/W	See section	15.2.11/360
588	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR210)	32	R/W	See section	15.2.11/360
58C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR211)	32	R/W	See section	15.2.11/360
590	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR212)	32	R/W	See section	15.2.11/360
594	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR213)	32	R/W	See section	15.2.11/360

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
598	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR214)	32	R/W	See section	15.2.11/360
59C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR215)	32	R/W	See section	15.2.11/360
5A0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR216)	32	R/W	See section	15.2.11/360
5A4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR217)	32	R/W	See section	15.2.11/360
5A8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR218)	32	R/W	See section	15.2.11/360
5AC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR219)	32	R/W	See section	15.2.11/360
5B0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR220)	32	R/W	See section	15.2.11/360
5B4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR221)	32	R/W	See section	15.2.11/360
5B8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR222)	32	R/W	See section	15.2.11/360
5BC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR223)	32	R/W	See section	15.2.11/360
5C0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR224)	32	R/W	See section	15.2.11/360
5C4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR225)	32	R/W	See section	15.2.11/360
5C8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR226)	32	R/W	See section	15.2.11/360
5CC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR227)	32	R/W	See section	15.2.11/360
5D0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR228)	32	R/W	See section	15.2.11/360
5D4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR229)	32	R/W	See section	15.2.11/360
5D8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR230)	32	R/W	See section	15.2.11/360
5DC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR231)	32	R/W	See section	15.2.11/360
5E0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR232)	32	R/W	See section	15.2.11/360
5E4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR233)	32	R/W	See section	15.2.11/360
5E8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR234)	32	R/W	See section	15.2.11/360
5EC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR235)	32	R/W	See section	15.2.11/360

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5F0	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR236)	32	R/W	See section	15.2.11/360
5F4	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR237)	32	R/W	See section	15.2.11/360
5F8	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR238)	32	R/W	See section	15.2.11/360
5FC	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR239)	32	R/W	See section	15.2.11/360
600	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR240)	32	R/W	See section	15.2.11/360
604	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR241)	32	R/W	See section	15.2.11/360
608	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR242)	32	R/W	See section	15.2.11/360
60C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR243)	32	R/W	See section	15.2.11/360
610	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR244)	32	R/W	See section	15.2.11/360
614	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR245)	32	R/W	See section	15.2.11/360
618	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR246)	32	R/W	See section	15.2.11/360
61C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR247)	32	R/W	See section	15.2.11/360
620	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR248)	32	R/W	See section	15.2.11/360
624	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR249)	32	R/W	See section	15.2.11/360
628	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR250)	32	R/W	See section	15.2.11/360
62C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR251)	32	R/W	See section	15.2.11/360
630	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR252)	32	R/W	See section	15.2.11/360
634	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR253)	32	R/W	See section	15.2.11/360
638	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR254)	32	R/W	See section	15.2.11/360
63C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR255)	32	R/W	See section	15.2.11/360
640	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR256)	32	R/W	See section	15.2.11/360
644	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR257)	32	R/W	See section	15.2.11/360

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
648	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR258)	32	R/W	See section	15.2.11/360
64C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR259)	32	R/W	See section	15.2.11/360
650	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR260)	32	R/W	See section	15.2.11/360
654	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR261)	32	R/W	See section	15.2.11/360
658	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR262)	32	R/W	See section	15.2.11/360
65C	SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR263)	32	R/W	See section	15.2.11/360
A40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR0)	32	R/W	See section	15.2.12/362
A44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR1)	32	R/W	See section	15.2.12/362
A48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR2)	32	R/W	See section	15.2.12/362
A4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR3)	32	R/W	See section	15.2.12/362
A50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR4)	32	R/W	See section	15.2.12/362
A54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR5)	32	R/W	See section	15.2.12/362
A58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR6)	32	R/W	See section	15.2.12/362
A5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR7)	32	R/W	See section	15.2.12/362
A60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR8)	32	R/W	See section	15.2.12/362
A64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR9)	32	R/W	See section	15.2.12/362
A68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR10)	32	R/W	See section	15.2.12/362
A6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR11)	32	R/W	See section	15.2.12/362
A70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR12)	32	R/W	See section	15.2.12/362
A74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR13)	32	R/W	See section	15.2.12/362
A78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR14)	32	R/W	See section	15.2.12/362
A7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR15)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR16)	32	R/W	See section	15.2.12/362
A84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR17)	32	R/W	See section	15.2.12/362
A88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR18)	32	R/W	See section	15.2.12/362
A8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR19)	32	R/W	See section	15.2.12/362
A90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR20)	32	R/W	See section	15.2.12/362
A94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR21)	32	R/W	See section	15.2.12/362
A98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR22)	32	R/W	See section	15.2.12/362
A9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR23)	32	R/W	See section	15.2.12/362
AA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR24)	32	R/W	See section	15.2.12/362
AA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR25)	32	R/W	See section	15.2.12/362
AA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR26)	32	R/W	See section	15.2.12/362
AAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR27)	32	R/W	See section	15.2.12/362
AB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR28)	32	R/W	See section	15.2.12/362
AB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR29)	32	R/W	See section	15.2.12/362
AB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR30)	32	R/W	See section	15.2.12/362
ABC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR31)	32	R/W	See section	15.2.12/362
AC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR32)	32	R/W	See section	15.2.12/362
AC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR33)	32	R/W	See section	15.2.12/362
AC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR34)	32	R/W	See section	15.2.12/362
ACC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR35)	32	R/W	See section	15.2.12/362
AD0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR36)	32	R/W	See section	15.2.12/362
AD4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR37)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AD8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR38)	32	R/W	See section	15.2.12/362
ADC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR39)	32	R/W	See section	15.2.12/362
AE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR40)	32	R/W	See section	15.2.12/362
AE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR41)	32	R/W	See section	15.2.12/362
AE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR42)	32	R/W	See section	15.2.12/362
AEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR43)	32	R/W	See section	15.2.12/362
AF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR44)	32	R/W	See section	15.2.12/362
AF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR45)	32	R/W	See section	15.2.12/362
AF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR46)	32	R/W	See section	15.2.12/362
AFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR47)	32	R/W	See section	15.2.12/362
B00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR48)	32	R/W	See section	15.2.12/362
B04	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR49)	32	R/W	See section	15.2.12/362
B08	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR50)	32	R/W	See section	15.2.12/362
B0C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR51)	32	R/W	See section	15.2.12/362
B10	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR52)	32	R/W	See section	15.2.12/362
B14	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR53)	32	R/W	See section	15.2.12/362
B18	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR54)	32	R/W	See section	15.2.12/362
B1C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR55)	32	R/W	See section	15.2.12/362
B20	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR56)	32	R/W	See section	15.2.12/362
B24	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR57)	32	R/W	See section	15.2.12/362
B28	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR58)	32	R/W	See section	15.2.12/362
B2C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR59)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B30	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR60)	32	R/W	See section	15.2.12/362
B34	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR61)	32	R/W	See section	15.2.12/362
B38	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR62)	32	R/W	See section	15.2.12/362
B3C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR63)	32	R/W	See section	15.2.12/362
B40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR64)	32	R/W	See section	15.2.12/362
B44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR65)	32	R/W	See section	15.2.12/362
B48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR66)	32	R/W	See section	15.2.12/362
B4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR67)	32	R/W	See section	15.2.12/362
B50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR68)	32	R/W	See section	15.2.12/362
B54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR69)	32	R/W	See section	15.2.12/362
B58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR70)	32	R/W	See section	15.2.12/362
B5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR71)	32	R/W	See section	15.2.12/362
B60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR72)	32	R/W	See section	15.2.12/362
B64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR73)	32	R/W	See section	15.2.12/362
B68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR74)	32	R/W	See section	15.2.12/362
B6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR75)	32	R/W	See section	15.2.12/362
B70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR76)	32	R/W	See section	15.2.12/362
B74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR77)	32	R/W	See section	15.2.12/362
B78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR78)	32	R/W	See section	15.2.12/362
B7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR79)	32	R/W	See section	15.2.12/362
B80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR80)	32	R/W	See section	15.2.12/362
B84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR81)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR82)	32	R/W	See section	15.2.12/362
B8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR83)	32	R/W	See section	15.2.12/362
B90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR84)	32	R/W	See section	15.2.12/362
B94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR85)	32	R/W	See section	15.2.12/362
B98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR86)	32	R/W	See section	15.2.12/362
B9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR87)	32	R/W	See section	15.2.12/362
BA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR88)	32	R/W	See section	15.2.12/362
BA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR89)	32	R/W	See section	15.2.12/362
BA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR90)	32	R/W	See section	15.2.12/362
BAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR91)	32	R/W	See section	15.2.12/362
BB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR92)	32	R/W	See section	15.2.12/362
BB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR93)	32	R/W	See section	15.2.12/362
BB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR94)	32	R/W	See section	15.2.12/362
BBC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR95)	32	R/W	See section	15.2.12/362
BC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR96)	32	R/W	See section	15.2.12/362
BC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR97)	32	R/W	See section	15.2.12/362
BC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR98)	32	R/W	See section	15.2.12/362
BCC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR99)	32	R/W	See section	15.2.12/362
BD0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR100)	32	R/W	See section	15.2.12/362
BD4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR101)	32	R/W	See section	15.2.12/362
BD8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR102)	32	R/W	See section	15.2.12/362
BDC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR103)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
BE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR104)	32	R/W	See section	15.2.12/362
BE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR105)	32	R/W	See section	15.2.12/362
BE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR106)	32	R/W	See section	15.2.12/362
BEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR107)	32	R/W	See section	15.2.12/362
BF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR108)	32	R/W	See section	15.2.12/362
BF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR109)	32	R/W	See section	15.2.12/362
BF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR110)	32	R/W	See section	15.2.12/362
BFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR111)	32	R/W	See section	15.2.12/362
C00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR112)	32	R/W	See section	15.2.12/362
C04	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR113)	32	R/W	See section	15.2.12/362
C08	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR114)	32	R/W	See section	15.2.12/362
C0C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR115)	32	R/W	See section	15.2.12/362
C10	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR116)	32	R/W	See section	15.2.12/362
C14	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR117)	32	R/W	See section	15.2.12/362
C18	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR118)	32	R/W	See section	15.2.12/362
C1C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR119)	32	R/W	See section	15.2.12/362
C20	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR120)	32	R/W	See section	15.2.12/362
C24	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR121)	32	R/W	See section	15.2.12/362
C28	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR122)	32	R/W	See section	15.2.12/362
C2C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR123)	32	R/W	See section	15.2.12/362
C30	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR124)	32	R/W	See section	15.2.12/362
C34	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR125)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C38	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR126)	32	R/W	See section	15.2.12/362
C3C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR127)	32	R/W	See section	15.2.12/362
C40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR128)	32	R/W	See section	15.2.12/362
C44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR129)	32	R/W	See section	15.2.12/362
C48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR130)	32	R/W	See section	15.2.12/362
C4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR131)	32	R/W	See section	15.2.12/362
C50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR132)	32	R/W	See section	15.2.12/362
C54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR133)	32	R/W	See section	15.2.12/362
C58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR134)	32	R/W	See section	15.2.12/362
C5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR135)	32	R/W	See section	15.2.12/362
C60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR136)	32	R/W	See section	15.2.12/362
C64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR137)	32	R/W	See section	15.2.12/362
C68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR138)	32	R/W	See section	15.2.12/362
C6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR139)	32	R/W	See section	15.2.12/362
C70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR140)	32	R/W	See section	15.2.12/362
C74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR141)	32	R/W	See section	15.2.12/362
C78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR142)	32	R/W	See section	15.2.12/362
C7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR143)	32	R/W	See section	15.2.12/362
C80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR144)	32	R/W	See section	15.2.12/362
C84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR145)	32	R/W	See section	15.2.12/362
C88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR146)	32	R/W	See section	15.2.12/362
C8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR147)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR148)	32	R/W	See section	15.2.12/362
C94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR149)	32	R/W	See section	15.2.12/362
C98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR150)	32	R/W	See section	15.2.12/362
C9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR151)	32	R/W	See section	15.2.12/362
CA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR152)	32	R/W	See section	15.2.12/362
CA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR153)	32	R/W	See section	15.2.12/362
CA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR154)	32	R/W	See section	15.2.12/362
CAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR155)	32	R/W	See section	15.2.12/362
CB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR156)	32	R/W	See section	15.2.12/362
CB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR157)	32	R/W	See section	15.2.12/362
CB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR158)	32	R/W	See section	15.2.12/362
CBC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR159)	32	R/W	See section	15.2.12/362
CC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR160)	32	R/W	See section	15.2.12/362
CC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR161)	32	R/W	See section	15.2.12/362
CC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR162)	32	R/W	See section	15.2.12/362
CCC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR163)	32	R/W	See section	15.2.12/362
CD0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR164)	32	R/W	See section	15.2.12/362
CD4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR165)	32	R/W	See section	15.2.12/362
CD8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR166)	32	R/W	See section	15.2.12/362
CDC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR167)	32	R/W	See section	15.2.12/362
CE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR168)	32	R/W	See section	15.2.12/362
CE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR169)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
CE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR170)	32	R/W	See section	15.2.12/362
CEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR171)	32	R/W	See section	15.2.12/362
CF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR172)	32	R/W	See section	15.2.12/362
CF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR173)	32	R/W	See section	15.2.12/362
CF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR174)	32	R/W	See section	15.2.12/362
CFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR175)	32	R/W	See section	15.2.12/362
D00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR176)	32	R/W	See section	15.2.12/362
D04	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR177)	32	R/W	See section	15.2.12/362
D08	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR178)	32	R/W	See section	15.2.12/362
D0C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR179)	32	R/W	See section	15.2.12/362
D10	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR180)	32	R/W	See section	15.2.12/362
D14	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR181)	32	R/W	See section	15.2.12/362
D18	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR182)	32	R/W	See section	15.2.12/362
D1C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR183)	32	R/W	See section	15.2.12/362
D20	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR184)	32	R/W	See section	15.2.12/362
D24	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR185)	32	R/W	See section	15.2.12/362
D28	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR186)	32	R/W	See section	15.2.12/362
D2C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR187)	32	R/W	See section	15.2.12/362
D30	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR188)	32	R/W	See section	15.2.12/362
D34	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR189)	32	R/W	See section	15.2.12/362
D38	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR190)	32	R/W	See section	15.2.12/362
D3C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR191)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR192)	32	R/W	See section	15.2.12/362
D44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR193)	32	R/W	See section	15.2.12/362
D48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR194)	32	R/W	See section	15.2.12/362
D4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR195)	32	R/W	See section	15.2.12/362
D50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR196)	32	R/W	See section	15.2.12/362
D54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR197)	32	R/W	See section	15.2.12/362
D58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR198)	32	R/W	See section	15.2.12/362
D5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR199)	32	R/W	See section	15.2.12/362
D60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR200)	32	R/W	See section	15.2.12/362
D64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR201)	32	R/W	See section	15.2.12/362
D68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR202)	32	R/W	See section	15.2.12/362
D6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR203)	32	R/W	See section	15.2.12/362
D70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR204)	32	R/W	See section	15.2.12/362
D74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR205)	32	R/W	See section	15.2.12/362
D78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR206)	32	R/W	See section	15.2.12/362
D7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR207)	32	R/W	See section	15.2.12/362
D80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR208)	32	R/W	See section	15.2.12/362
D84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR209)	32	R/W	See section	15.2.12/362
D88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR210)	32	R/W	See section	15.2.12/362
D8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR211)	32	R/W	See section	15.2.12/362
D90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR212)	32	R/W	See section	15.2.12/362
D94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR213)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR214)	32	R/W	See section	15.2.12/362
D9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR215)	32	R/W	See section	15.2.12/362
DA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR216)	32	R/W	See section	15.2.12/362
DA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR217)	32	R/W	See section	15.2.12/362
DA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR218)	32	R/W	See section	15.2.12/362
DAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR219)	32	R/W	See section	15.2.12/362
DB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR220)	32	R/W	See section	15.2.12/362
DB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR221)	32	R/W	See section	15.2.12/362
DB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR222)	32	R/W	See section	15.2.12/362
DBC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR223)	32	R/W	See section	15.2.12/362
DC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR224)	32	R/W	See section	15.2.12/362
DC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR225)	32	R/W	See section	15.2.12/362
DC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR226)	32	R/W	See section	15.2.12/362
DCC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR227)	32	R/W	See section	15.2.12/362
DD0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR228)	32	R/W	See section	15.2.12/362
DD4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR229)	32	R/W	See section	15.2.12/362
DD8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR230)	32	R/W	See section	15.2.12/362
DDC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR231)	32	R/W	See section	15.2.12/362
DE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR232)	32	R/W	See section	15.2.12/362
DE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR233)	32	R/W	See section	15.2.12/362
DE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR234)	32	R/W	See section	15.2.12/362
DEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR235)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
DF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR236)	32	R/W	See section	15.2.12/362
DF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR237)	32	R/W	See section	15.2.12/362
DF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR238)	32	R/W	See section	15.2.12/362
DFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR239)	32	R/W	See section	15.2.12/362
E00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR240)	32	R/W	See section	15.2.12/362
E04	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR241)	32	R/W	See section	15.2.12/362
E08	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR242)	32	R/W	See section	15.2.12/362
E0C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR243)	32	R/W	See section	15.2.12/362
E10	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR244)	32	R/W	See section	15.2.12/362
E14	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR245)	32	R/W	See section	15.2.12/362
E18	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR246)	32	R/W	See section	15.2.12/362
E1C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR247)	32	R/W	See section	15.2.12/362
E20	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR248)	32	R/W	See section	15.2.12/362
E24	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR249)	32	R/W	See section	15.2.12/362
E28	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR250)	32	R/W	See section	15.2.12/362
E2C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR251)	32	R/W	See section	15.2.12/362
E30	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR252)	32	R/W	See section	15.2.12/362
E34	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR253)	32	R/W	See section	15.2.12/362
E38	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR254)	32	R/W	See section	15.2.12/362
E3C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR255)	32	R/W	See section	15.2.12/362
E40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR256)	32	R/W	See section	15.2.12/362
E44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR257)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR258)	32	R/W	See section	15.2.12/362
E4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR259)	32	R/W	See section	15.2.12/362
E50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR260)	32	R/W	See section	15.2.12/362
E54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR261)	32	R/W	See section	15.2.12/362
E58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR262)	32	R/W	See section	15.2.12/362
E5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR263)	32	R/W	See section	15.2.12/362
E60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR264)	32	R/W	See section	15.2.12/362
E64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR265)	32	R/W	See section	15.2.12/362
E68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR266)	32	R/W	See section	15.2.12/362
E6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR267)	32	R/W	See section	15.2.12/362
E70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR268)	32	R/W	See section	15.2.12/362
E74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR269)	32	R/W	See section	15.2.12/362
E78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR270)	32	R/W	See section	15.2.12/362
E7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR271)	32	R/W	See section	15.2.12/362
E80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR272)	32	R/W	See section	15.2.12/362
E84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR273)	32	R/W	See section	15.2.12/362
E88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR274)	32	R/W	See section	15.2.12/362
E8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR275)	32	R/W	See section	15.2.12/362
E90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR276)	32	R/W	See section	15.2.12/362
E94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR277)	32	R/W	See section	15.2.12/362
E98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR278)	32	R/W	See section	15.2.12/362
E9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR279)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
EA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR280)	32	R/W	See section	15.2.12/362
EA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR281)	32	R/W	See section	15.2.12/362
EA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR282)	32	R/W	See section	15.2.12/362
EAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR283)	32	R/W	See section	15.2.12/362
EB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR284)	32	R/W	See section	15.2.12/362
EB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR285)	32	R/W	See section	15.2.12/362
EB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR286)	32	R/W	See section	15.2.12/362
EBC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR287)	32	R/W	See section	15.2.12/362
EC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR288)	32	R/W	See section	15.2.12/362
EC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR289)	32	R/W	See section	15.2.12/362
EC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR290)	32	R/W	See section	15.2.12/362
ECC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR291)	32	R/W	See section	15.2.12/362
ED0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR292)	32	R/W	See section	15.2.12/362
ED4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR293)	32	R/W	See section	15.2.12/362
ED8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR294)	32	R/W	See section	15.2.12/362
EDC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR295)	32	R/W	See section	15.2.12/362
EE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR296)	32	R/W	See section	15.2.12/362
EE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR297)	32	R/W	See section	15.2.12/362
EE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR298)	32	R/W	See section	15.2.12/362
EEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR299)	32	R/W	See section	15.2.12/362
EF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR300)	32	R/W	See section	15.2.12/362
EF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR301)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
EF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR302)	32	R/W	See section	15.2.12/362
EFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR303)	32	R/W	See section	15.2.12/362
F00	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR304)	32	R/W	See section	15.2.12/362
F04	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR305)	32	R/W	See section	15.2.12/362
F08	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR306)	32	R/W	See section	15.2.12/362
F0C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR307)	32	R/W	See section	15.2.12/362
F10	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR308)	32	R/W	See section	15.2.12/362
F14	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR309)	32	R/W	See section	15.2.12/362
F18	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR310)	32	R/W	See section	15.2.12/362
F1C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR311)	32	R/W	See section	15.2.12/362
F20	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR312)	32	R/W	See section	15.2.12/362
F24	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR313)	32	R/W	See section	15.2.12/362
F28	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR314)	32	R/W	See section	15.2.12/362
F2C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR315)	32	R/W	See section	15.2.12/362
F30	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR316)	32	R/W	See section	15.2.12/362
F34	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR317)	32	R/W	See section	15.2.12/362
F38	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR318)	32	R/W	See section	15.2.12/362
F3C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR319)	32	R/W	See section	15.2.12/362
F40	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR320)	32	R/W	See section	15.2.12/362
F44	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR321)	32	R/W	See section	15.2.12/362
F48	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR322)	32	R/W	See section	15.2.12/362
F4C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR323)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F50	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR324)	32	R/W	See section	15.2.12/362
F54	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR325)	32	R/W	See section	15.2.12/362
F58	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR326)	32	R/W	See section	15.2.12/362
F5C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR327)	32	R/W	See section	15.2.12/362
F60	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR328)	32	R/W	See section	15.2.12/362
F64	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR329)	32	R/W	See section	15.2.12/362
F68	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR330)	32	R/W	See section	15.2.12/362
F6C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR331)	32	R/W	See section	15.2.12/362
F70	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR332)	32	R/W	See section	15.2.12/362
F74	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR333)	32	R/W	See section	15.2.12/362
F78	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR334)	32	R/W	See section	15.2.12/362
F7C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR335)	32	R/W	See section	15.2.12/362
F80	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR336)	32	R/W	See section	15.2.12/362
F84	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR337)	32	R/W	See section	15.2.12/362
F88	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR338)	32	R/W	See section	15.2.12/362
F8C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR339)	32	R/W	See section	15.2.12/362
F90	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR340)	32	R/W	See section	15.2.12/362
F94	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR341)	32	R/W	See section	15.2.12/362
F98	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR342)	32	R/W	See section	15.2.12/362
F9C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR343)	32	R/W	See section	15.2.12/362
FA0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR344)	32	R/W	See section	15.2.12/362
FA4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR345)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FA8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR346)	32	R/W	See section	15.2.12/362
FAC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR347)	32	R/W	See section	15.2.12/362
FB0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR348)	32	R/W	See section	15.2.12/362
FB4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR349)	32	R/W	See section	15.2.12/362
FB8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR350)	32	R/W	See section	15.2.12/362
FBC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR351)	32	R/W	See section	15.2.12/362
FC0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR352)	32	R/W	See section	15.2.12/362
FC4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR353)	32	R/W	See section	15.2.12/362
FC8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR354)	32	R/W	See section	15.2.12/362
FCC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR355)	32	R/W	See section	15.2.12/362
FD0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR356)	32	R/W	See section	15.2.12/362
FD4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR357)	32	R/W	See section	15.2.12/362
FD8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR358)	32	R/W	See section	15.2.12/362
FDC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR359)	32	R/W	See section	15.2.12/362
FE0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR360)	32	R/W	See section	15.2.12/362
FE4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR361)	32	R/W	See section	15.2.12/362
FE8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR362)	32	R/W	See section	15.2.12/362
FEC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR363)	32	R/W	See section	15.2.12/362
FF0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR364)	32	R/W	See section	15.2.12/362
FF4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR365)	32	R/W	See section	15.2.12/362
FF8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR366)	32	R/W	See section	15.2.12/362
FFC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR367)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1000	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR368)	32	R/W	See section	15.2.12/362
1004	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR369)	32	R/W	See section	15.2.12/362
1008	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR370)	32	R/W	See section	15.2.12/362
100C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR371)	32	R/W	See section	15.2.12/362
1010	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR372)	32	R/W	See section	15.2.12/362
1014	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR373)	32	R/W	See section	15.2.12/362
1018	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR374)	32	R/W	See section	15.2.12/362
101C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR375)	32	R/W	See section	15.2.12/362
1020	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR376)	32	R/W	See section	15.2.12/362
1024	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR377)	32	R/W	See section	15.2.12/362
1028	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR378)	32	R/W	See section	15.2.12/362
102C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR379)	32	R/W	See section	15.2.12/362
1030	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR380)	32	R/W	See section	15.2.12/362
1034	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR381)	32	R/W	See section	15.2.12/362
1038	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR382)	32	R/W	See section	15.2.12/362
103C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR383)	32	R/W	See section	15.2.12/362
1040	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR384)	32	R/W	See section	15.2.12/362
1044	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR385)	32	R/W	See section	15.2.12/362
1048	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR386)	32	R/W	See section	15.2.12/362
104C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR387)	32	R/W	See section	15.2.12/362
1050	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR388)	32	R/W	See section	15.2.12/362
1054	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR389)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1058	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR390)	32	R/W	See section	15.2.12/362
105C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR391)	32	R/W	See section	15.2.12/362
1060	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR392)	32	R/W	See section	15.2.12/362
1064	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR393)	32	R/W	See section	15.2.12/362
1068	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR394)	32	R/W	See section	15.2.12/362
106C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR395)	32	R/W	See section	15.2.12/362
1070	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR396)	32	R/W	See section	15.2.12/362
1074	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR397)	32	R/W	See section	15.2.12/362
1078	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR398)	32	R/W	See section	15.2.12/362
107C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR399)	32	R/W	See section	15.2.12/362
1080	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR400)	32	R/W	See section	15.2.12/362
1084	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR401)	32	R/W	See section	15.2.12/362
1088	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR402)	32	R/W	See section	15.2.12/362
108C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR403)	32	R/W	See section	15.2.12/362
1090	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR404)	32	R/W	See section	15.2.12/362
1094	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR405)	32	R/W	See section	15.2.12/362
1098	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR406)	32	R/W	See section	15.2.12/362
109C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR407)	32	R/W	See section	15.2.12/362
10A0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR408)	32	R/W	See section	15.2.12/362
10A4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR409)	32	R/W	See section	15.2.12/362
10A8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR410)	32	R/W	See section	15.2.12/362
10AC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR411)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10B0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR412)	32	R/W	See section	15.2.12/362
10B4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR413)	32	R/W	See section	15.2.12/362
10B8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR414)	32	R/W	See section	15.2.12/362
10BC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR415)	32	R/W	See section	15.2.12/362
10C0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR416)	32	R/W	See section	15.2.12/362
10C4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR417)	32	R/W	See section	15.2.12/362
10C8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR418)	32	R/W	See section	15.2.12/362
10CC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR419)	32	R/W	See section	15.2.12/362
10D0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR420)	32	R/W	See section	15.2.12/362
10D4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR421)	32	R/W	See section	15.2.12/362
10D8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR422)	32	R/W	See section	15.2.12/362
10DC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR423)	32	R/W	See section	15.2.12/362
10E0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR424)	32	R/W	See section	15.2.12/362
10E4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR425)	32	R/W	See section	15.2.12/362
10E8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR426)	32	R/W	See section	15.2.12/362
10EC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR427)	32	R/W	See section	15.2.12/362
10F0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR428)	32	R/W	See section	15.2.12/362
10F4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR429)	32	R/W	See section	15.2.12/362
10F8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR430)	32	R/W	See section	15.2.12/362
10FC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR431)	32	R/W	See section	15.2.12/362
1100	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR432)	32	R/W	See section	15.2.12/362
1104	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR433)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1108	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR434)	32	R/W	See section	15.2.12/362
110C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR435)	32	R/W	See section	15.2.12/362
1110	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR436)	32	R/W	See section	15.2.12/362
1114	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR437)	32	R/W	See section	15.2.12/362
1118	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR438)	32	R/W	See section	15.2.12/362
111C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR439)	32	R/W	See section	15.2.12/362
1120	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR440)	32	R/W	See section	15.2.12/362
1124	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR441)	32	R/W	See section	15.2.12/362
1128	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR442)	32	R/W	See section	15.2.12/362
112C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR443)	32	R/W	See section	15.2.12/362
1130	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR444)	32	R/W	See section	15.2.12/362
1134	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR445)	32	R/W	See section	15.2.12/362
1138	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR446)	32	R/W	See section	15.2.12/362
113C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR447)	32	R/W	See section	15.2.12/362
1140	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR448)	32	R/W	See section	15.2.12/362
1144	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR449)	32	R/W	See section	15.2.12/362
1148	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR450)	32	R/W	See section	15.2.12/362
114C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR451)	32	R/W	See section	15.2.12/362
1150	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR452)	32	R/W	See section	15.2.12/362
1154	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR453)	32	R/W	See section	15.2.12/362
1158	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR454)	32	R/W	See section	15.2.12/362
115C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR455)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1160	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR456)	32	R/W	See section	15.2.12/362
1164	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR457)	32	R/W	See section	15.2.12/362
1168	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR458)	32	R/W	See section	15.2.12/362
116C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR459)	32	R/W	See section	15.2.12/362
1170	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR460)	32	R/W	See section	15.2.12/362
1174	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR461)	32	R/W	See section	15.2.12/362
1178	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR462)	32	R/W	See section	15.2.12/362
117C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR463)	32	R/W	See section	15.2.12/362
1180	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR464)	32	R/W	See section	15.2.12/362
1184	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR465)	32	R/W	See section	15.2.12/362
1188	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR466)	32	R/W	See section	15.2.12/362
118C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR467)	32	R/W	See section	15.2.12/362
1190	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR468)	32	R/W	See section	15.2.12/362
1194	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR469)	32	R/W	See section	15.2.12/362
1198	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR470)	32	R/W	See section	15.2.12/362
119C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR471)	32	R/W	See section	15.2.12/362
11A0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR472)	32	R/W	See section	15.2.12/362
11A4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR473)	32	R/W	See section	15.2.12/362
11A8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR474)	32	R/W	See section	15.2.12/362
11AC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR475)	32	R/W	See section	15.2.12/362
11B0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR476)	32	R/W	See section	15.2.12/362
11B4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR477)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11B8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR478)	32	R/W	See section	15.2.12/362
11BC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR479)	32	R/W	See section	15.2.12/362
11C0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR480)	32	R/W	See section	15.2.12/362
11C4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR481)	32	R/W	See section	15.2.12/362
11C8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR482)	32	R/W	See section	15.2.12/362
11CC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR483)	32	R/W	See section	15.2.12/362
11D0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR484)	32	R/W	See section	15.2.12/362
11D4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR485)	32	R/W	See section	15.2.12/362
11D8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR486)	32	R/W	See section	15.2.12/362
11DC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR487)	32	R/W	See section	15.2.12/362
11E0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR488)	32	R/W	See section	15.2.12/362
11E4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR489)	32	R/W	See section	15.2.12/362
11E8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR490)	32	R/W	See section	15.2.12/362
11EC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR491)	32	R/W	See section	15.2.12/362
11F0	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR492)	32	R/W	See section	15.2.12/362
11F4	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR493)	32	R/W	See section	15.2.12/362
11F8	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR494)	32	R/W	See section	15.2.12/362
11FC	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR495)	32	R/W	See section	15.2.12/362
1200	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR496)	32	R/W	See section	15.2.12/362
1204	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR497)	32	R/W	See section	15.2.12/362
1208	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR498)	32	R/W	See section	15.2.12/362
120C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR499)	32	R/W	See section	15.2.12/362

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1210	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR500)	32	R/W	See section	15.2.12/362
1214	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR501)	32	R/W	See section	15.2.12/362
1218	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR502)	32	R/W	See section	15.2.12/362
121C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR503)	32	R/W	See section	15.2.12/362
1220	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR504)	32	R/W	See section	15.2.12/362
1224	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR505)	32	R/W	See section	15.2.12/362
1228	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR506)	32	R/W	See section	15.2.12/362
122C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR507)	32	R/W	See section	15.2.12/362
1230	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR508)	32	R/W	See section	15.2.12/362
1234	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR509)	32	R/W	See section	15.2.12/362
1238	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR510)	32	R/W	See section	15.2.12/362
123C	SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR511)	32	R/W	See section	15.2.12/362
1300	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO0_3)	32	R/W	0000_0000h	15.2.13/365
1304	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO4_7)	32	R/W	0000_0000h	15.2.13/365
1308	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO8_11)	32	R/W	0000_0000h	15.2.13/365
130C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO12_15)	32	R/W	0000_0000h	15.2.13/365
1310	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO16_19)	32	R/W	0000_0000h	15.2.13/365
1314	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO20_23)	32	R/W	0000_0000h	15.2.13/365
1318	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO24_27)	32	R/W	0000_0000h	15.2.13/365
131C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO28_31)	32	R/W	0000_0000h	15.2.13/365
1320	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO32_35)	32	R/W	0000_0000h	15.2.13/365
1324	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO36_39)	32	R/W	0000_0000h	15.2.13/365
1328	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO40_43)	32	R/W	0000_0000h	15.2.13/365
132C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO44_47)	32	R/W	0000_0000h	15.2.13/365

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1330	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO48_51)	32	R/W	0000_0000h	15.2.13/365
1334	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO52_55)	32	R/W	0000_0000h	15.2.13/365
1338	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO56_59)	32	R/W	0000_0000h	15.2.13/365
133C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO60_63)	32	R/W	0000_0000h	15.2.13/365
1340	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO64_67)	32	R/W	0000_0000h	15.2.13/365
1344	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO68_71)	32	R/W	0000_0000h	15.2.13/365
1348	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO72_75)	32	R/W	0000_0000h	15.2.13/365
134C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO76_79)	32	R/W	0000_0000h	15.2.13/365
1350	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO80_83)	32	R/W	0000_0000h	15.2.13/365
1354	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO84_87)	32	R/W	0000_0000h	15.2.13/365
1358	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO88_91)	32	R/W	0000_0000h	15.2.13/365
135C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO92_95)	32	R/W	0000_0000h	15.2.13/365
1360	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO96_99)	32	R/W	0000_0000h	15.2.13/365
1364	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO100_103)	32	R/W	0000_0000h	15.2.13/365
1368	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO104_107)	32	R/W	0000_0000h	15.2.13/365
136C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO108_111)	32	R/W	0000_0000h	15.2.13/365
1370	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO112_115)	32	R/W	0000_0000h	15.2.13/365
1374	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO116_119)	32	R/W	0000_0000h	15.2.13/365
1378	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO120_123)	32	R/W	0000_0000h	15.2.13/365
137C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO124_127)	32	R/W	0000_0000h	15.2.13/365
1380	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO128_131)	32	R/W	0000_0000h	15.2.13/365
1384	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO132_135)	32	R/W	0000_0000h	15.2.13/365

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1388	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO136_139)	32	R/W	0000_0000h	15.2.13/365
138C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO140_143)	32	R/W	0000_0000h	15.2.13/365
1390	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO144_147)	32	R/W	0000_0000h	15.2.13/365
1394	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO148_151)	32	R/W	0000_0000h	15.2.13/365
1398	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO152_155)	32	R/W	0000_0000h	15.2.13/365
139C	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO156_159)	32	R/W	0000_0000h	15.2.13/365
13A0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO160_163)	32	R/W	0000_0000h	15.2.13/365
13A4	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO164_167)	32	R/W	0000_0000h	15.2.13/365
13A8	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO168_171)	32	R/W	0000_0000h	15.2.13/365
13AC	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO172_175)	32	R/W	0000_0000h	15.2.13/365
13B0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO176_179)	32	R/W	0000_0000h	15.2.13/365
13B4	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO180_183)	32	R/W	0000_0000h	15.2.13/365
13B8	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO184_187)	32	R/W	0000_0000h	15.2.13/365
13BC	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO188_191)	32	R/W	0000_0000h	15.2.13/365
13C0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO192_195)	32	R/W	0000_0000h	15.2.13/365
13C4	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO196_199)	32	R/W	0000_0000h	15.2.13/365
13C8	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO200_203)	32	R/W	0000_0000h	15.2.13/365
13CC	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO204_207)	32	R/W	0000_0000h	15.2.13/365
13D0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO208_211)	32	R/W	0000_0000h	15.2.13/365
13D4	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO212_215)	32	R/W	0000_0000h	15.2.13/365
13D8	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO216_219)	32	R/W	0000_0000h	15.2.13/365
13DC	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO220_223)	32	R/W	0000_0000h	15.2.13/365

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13E0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO224_227)	32	R/W	0000_0000h	15.2.13/365
13E4	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO228_231)	32	R/W	0000_0000h	15.2.13/365
13E8	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO232_235)	32	R/W	0000_0000h	15.2.13/365
13EC	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO236_239)	32	R/W	0000_0000h	15.2.13/365
13F0	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO240_243)	32	R/W	0000_0000h	15.2.13/365
13F4	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO244_247)	32	R/W	0000_0000h	15.2.13/365
13F8	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO248_251)	32	R/W	0000_0000h	15.2.13/365
13FC	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO252_255)	32	R/W	0000_0000h	15.2.13/365
1400	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO256_259)	32	R/W	0000_0000h	15.2.13/365
1404	SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO260_263)	32	R/W	0000_0000h	15.2.13/365
1500	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI0_3)	32	R	0000_0000h	15.2.14/366
1504	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI4_7)	32	R	0000_0000h	15.2.14/366
1508	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI8_11)	32	R	0000_0000h	15.2.14/366
150C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI12_15)	32	R	0000_0000h	15.2.14/366
1510	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI16_19)	32	R	0000_0000h	15.2.14/366
1514	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI20_23)	32	R	0000_0000h	15.2.14/366
1518	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI24_27)	32	R	0000_0000h	15.2.14/366
151C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI28_31)	32	R	0000_0000h	15.2.14/366
1520	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI32_35)	32	R	0000_0000h	15.2.14/366
1524	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI36_39)	32	R	0000_0000h	15.2.14/366
1528	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI40_43)	32	R	0000_0000h	15.2.14/366
152C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI44_47)	32	R	0000_0000h	15.2.14/366
1530	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI48_51)	32	R	0000_0000h	15.2.14/366
1534	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI52_55)	32	R	0000_0000h	15.2.14/366
1538	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI56_59)	32	R	0000_0000h	15.2.14/366
153C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI60_63)	32	R	0000_0000h	15.2.14/366
1540	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI64_67)	32	R	0000_0000h	15.2.14/366
1544	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI68_71)	32	R	0000_0000h	15.2.14/366
1548	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI72_75)	32	R	0000_0000h	15.2.14/366
154C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI76_79)	32	R	0000_0000h	15.2.14/366
1550	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI80_83)	32	R	0000_0000h	15.2.14/366

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1554	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI84_87)	32	R	0000_0000h	15.2.14/366
1558	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI88_91)	32	R	0000_0000h	15.2.14/366
155C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI92_95)	32	R	0000_0000h	15.2.14/366
1560	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI96_99)	32	R	0000_0000h	15.2.14/366
1564	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI100_103)	32	R	0000_0000h	15.2.14/366
1568	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI104_107)	32	R	0000_0000h	15.2.14/366
156C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI108_111)	32	R	0000_0000h	15.2.14/366
1570	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI112_115)	32	R	0000_0000h	15.2.14/366
1574	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI116_119)	32	R	0000_0000h	15.2.14/366
1578	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI120_123)	32	R	0000_0000h	15.2.14/366
157C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI124_127)	32	R	0000_0000h	15.2.14/366
1580	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI128_131)	32	R	0000_0000h	15.2.14/366
1584	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI132_135)	32	R	0000_0000h	15.2.14/366
1588	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI136_139)	32	R	0000_0000h	15.2.14/366
158C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI140_143)	32	R	0000_0000h	15.2.14/366
1590	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI144_147)	32	R	0000_0000h	15.2.14/366
1594	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI148_151)	32	R	0000_0000h	15.2.14/366
1598	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI152_155)	32	R	0000_0000h	15.2.14/366
159C	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI156_159)	32	R	0000_0000h	15.2.14/366
15A0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI160_163)	32	R	0000_0000h	15.2.14/366
15A4	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI164_167)	32	R	0000_0000h	15.2.14/366
15A8	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI168_171)	32	R	0000_0000h	15.2.14/366
15AC	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI172_175)	32	R	0000_0000h	15.2.14/366
15B0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI176_179)	32	R	0000_0000h	15.2.14/366

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
15B4	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI180_183)	32	R	0000_0000h	15.2.14/366
15B8	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI184_187)	32	R	0000_0000h	15.2.14/366
15BC	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI188_191)	32	R	0000_0000h	15.2.14/366
15C0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI192_195)	32	R	0000_0000h	15.2.14/366
15C4	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI196_199)	32	R	0000_0000h	15.2.14/366
15C8	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI200_203)	32	R	0000_0000h	15.2.14/366
15CC	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI204_207)	32	R	0000_0000h	15.2.14/366
15D0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI208_211)	32	R	0000_0000h	15.2.14/366
15D4	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI212_215)	32	R	0000_0000h	15.2.14/366
15D8	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI216_219)	32	R	0000_0000h	15.2.14/366
15DC	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI220_223)	32	R	0000_0000h	15.2.14/366
15E0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI224_227)	32	R	0000_0000h	15.2.14/366
15E4	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI228_231)	32	R	0000_0000h	15.2.14/366
15E8	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI232_235)	32	R	0000_0000h	15.2.14/366
15EC	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI236_239)	32	R	0000_0000h	15.2.14/366
15F0	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI240_243)	32	R	0000_0000h	15.2.14/366
15F4	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI244_247)	32	R	0000_0000h	15.2.14/366
15F8	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI248_251)	32	R	0000_0000h	15.2.14/366
15FC	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI252_255)	32	R	0000_0000h	15.2.14/366
1600	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI256_259)	32	R	0000_0000h	15.2.14/366
1604	SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI260_263)	32	R	0000_0000h	15.2.14/366
1700	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO0)	16	R/W	0000h	15.2.15/368

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1702	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO1)	16	R/W	0000h	15.2.15/368
1704	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO2)	16	R/W	0000h	15.2.15/368
1706	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO3)	16	R/W	0000h	15.2.15/368
1708	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO4)	16	R/W	0000h	15.2.15/368
170A	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO5)	16	R/W	0000h	15.2.15/368
170C	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO6)	16	R/W	0000h	15.2.15/368
170E	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO7)	16	R/W	0000h	15.2.15/368
1710	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO8)	16	R/W	0000h	15.2.15/368
1712	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO9)	16	R/W	0000h	15.2.15/368
1714	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO10)	16	R/W	0000h	15.2.15/368
1716	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO11)	16	R/W	0000h	15.2.15/368
1718	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO12)	16	R/W	0000h	15.2.15/368
171A	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO13)	16	R/W	0000h	15.2.15/368
171C	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO14)	16	R/W	0000h	15.2.15/368
171E	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO15)	16	R/W	0000h	15.2.15/368
1720	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO16)	16	R/W	0000h	15.2.15/368
1722	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO17)	16	R/W	0000h	15.2.15/368
1724	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO18)	16	R/W	0000h	15.2.15/368
1726	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO19)	16	R/W	0000h	15.2.15/368
1728	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO20)	16	R/W	0000h	15.2.15/368
172A	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO21)	16	R/W	0000h	15.2.15/368
172C	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO22)	16	R/W	0000h	15.2.15/368

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
172E	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO23)	16	R/W	0000h	15.2.15/368
1730	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO24)	16	R/W	0000h	15.2.15/368
1732	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO25)	16	R/W	0000h	15.2.15/368
1734	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO26)	16	R/W	0000h	15.2.15/368
1736	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO27)	16	R/W	0000h	15.2.15/368
1738	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO28)	16	R/W	0000h	15.2.15/368
173A	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO29)	16	R/W	0000h	15.2.15/368
173C	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO30)	16	R/W	0000h	15.2.15/368
173E	SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO31)	16	R/W	0000h	15.2.15/368
1740	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI0)	16	R	0000h	15.2.16/369
1742	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI1)	16	R	0000h	15.2.16/369
1744	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI2)	16	R	0000h	15.2.16/369
1746	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI3)	16	R	0000h	15.2.16/369
1748	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI4)	16	R	0000h	15.2.16/369
174A	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI5)	16	R	0000h	15.2.16/369
174C	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI6)	16	R	0000h	15.2.16/369
174E	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI7)	16	R	0000h	15.2.16/369
1750	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI8)	16	R	0000h	15.2.16/369
1752	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI9)	16	R	0000h	15.2.16/369
1754	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI10)	16	R	0000h	15.2.16/369
1756	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI11)	16	R	0000h	15.2.16/369
1758	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI12)	16	R	0000h	15.2.16/369
175A	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI13)	16	R	0000h	15.2.16/369
175C	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI14)	16	R	0000h	15.2.16/369
175E	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI15)	16	R	0000h	15.2.16/369
1760	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI16)	16	R	0000h	15.2.16/369
1762	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI17)	16	R	0000h	15.2.16/369

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1764	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI18)	16	R	0000h	15.2.16/369
1766	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI19)	16	R	0000h	15.2.16/369
1768	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI20)	16	R	0000h	15.2.16/369
176A	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI21)	16	R	0000h	15.2.16/369
176C	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI22)	16	R	0000h	15.2.16/369
176E	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI23)	16	R	0000h	15.2.16/369
1770	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI24)	16	R	0000h	15.2.16/369
1772	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI25)	16	R	0000h	15.2.16/369
1774	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI26)	16	R	0000h	15.2.16/369
1776	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI27)	16	R	0000h	15.2.16/369
1778	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI28)	16	R	0000h	15.2.16/369
177A	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI29)	16	R	0000h	15.2.16/369
177C	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI30)	16	R	0000h	15.2.16/369
177E	SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI31)	16	R	0000h	15.2.16/369
1780	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO0)	32	W	0000_0000h	15.2.17/369
1784	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO1)	32	W	0000_0000h	15.2.17/369
1788	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO2)	32	W	0000_0000h	15.2.17/369
178C	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO3)	32	W	0000_0000h	15.2.17/369
1790	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO4)	32	W	0000_0000h	15.2.17/369
1794	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO5)	32	W	0000_0000h	15.2.17/369
1798	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO6)	32	W	0000_0000h	15.2.17/369
179C	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO7)	32	W	0000_0000h	15.2.17/369

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
17A0	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO8)	32	W	0000_0000h	15.2.17/369
17A4	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO9)	32	W	0000_0000h	15.2.17/369
17A8	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO10)	32	W	0000_0000h	15.2.17/369
17AC	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO11)	32	W	0000_0000h	15.2.17/369
17B0	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO12)	32	W	0000_0000h	15.2.17/369
17B4	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO13)	32	W	0000_0000h	15.2.17/369
17B8	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO14)	32	W	0000_0000h	15.2.17/369
17BC	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO15)	32	W	0000_0000h	15.2.17/369
17C0	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO16)	32	W	0000_0000h	15.2.17/369
17C4	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO17)	32	W	0000_0000h	15.2.17/369
17C8	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO18)	32	W	0000_0000h	15.2.17/369
17CC	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO19)	32	W	0000_0000h	15.2.17/369
17D0	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO20)	32	W	0000_0000h	15.2.17/369
17D4	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO21)	32	W	0000_0000h	15.2.17/369
17D8	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO22)	32	W	0000_0000h	15.2.17/369
17DC	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO23)	32	W	0000_0000h	15.2.17/369
17E0	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO24)	32	W	0000_0000h	15.2.17/369
17E4	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO25)	32	W	0000_0000h	15.2.17/369
17E8	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO26)	32	W	0000_0000h	15.2.17/369
17EC	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO27)	32	W	0000_0000h	15.2.17/369
17F0	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO28)	32	W	0000_0000h	15.2.17/369
17F4	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO29)	32	W	0000_0000h	15.2.17/369

Table continues on the next page...

SIUL2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
17F8	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO30)	32	W	0000_0000h	15.2.17/369
17FC	SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO31)	32	W	0000_0000h	15.2.17/369

15.2.1 SIUL2 MCU ID Register #1 (SIUL2_MIDR1)

This register holds identification information about the device.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PARTNUM															
W																
Reset	0	1	0	1	0	1	1	1	0	1	0	0	0	1	1	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	PKG					0		MAJOR_MASK				MINOR_MASK			
W																
Reset	0	*	*	*	*	*	0	0	*	*	*	*	*	*	*	*

* Notes:
- MINOR_MASK field: * indicates Value is set at factory and cannot be changed.
- MAJOR_MASK field: * indicates Value is set at factory and cannot be changed.
- PKG field: * indicates Value is set at factory and cannot be changed.

SIUL2_MIDR1 field descriptions

Field	Description
0–15 PARTNUM	MCU Family 5746 For the full part number this field must be combined with MIDR2.PARTNUM[23:16] Value is set at factory and cannot be changed.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–21 PKG	Package Settings Can be read by software to determine the package type that is used for the particular device: 10100 : 324-ball BGA 01100 : 256-ball BGA 10001 : 176-pin QFP 01000 : 100-ball BGA Value is set at factory and cannot be changed

Table continues on the next page...

SIUL2_MIDR1 field descriptions (continued)

Field	Description
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 MAJOR_MASK	Major Mask Revision Value is set at factory and increments with each revision of device.
28–31 MINOR_MASK	Minor Mask Revision Value is set at factory and increments with each revision of device.

15.2.2 SIUL2 MCU ID Register #2 (SIUL2_MIDR2)

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	FLASH_SIZE_1				FLASH_SIZE_2			0							
W																
Reset	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PARTNUM								0							
W																
Reset	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0

SIUL2_MIDR2 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–4 FLASH_SIZE_1	Maximum Flash size of the MCU family (part1) Total flash memory size = FLASH_SIZE_1 + FLASH_SIZE_2: 0111 2M Value is set at factory and cannot be changed. All other bit combinations are reserved.
5–8 FLASH_SIZE_2	Maximum Flash size of the MCU family (part2) Total flash memory size = FLASH_SIZE_1 + FLASH_SIZE_2: 0100 1M Value is set at factory and cannot be changed. All other bit combinations are reserved.
9–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 PARTNUM	ASCII character in MCU Family

Table continues on the next page...

SIUL2_MIDR2 field descriptions (continued)

Field	Description
	This field specifies the part number suffix, and needs to be combined with MIDR1[PARTNUM] to indicate the MCU Family. All other values reserved for future use. Value is set at factory and cannot be changed. 01000011 Character "C"
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

15.2.3 SIUL2 Interrupt Status Flag Register0 (SIUL2_ISR0)

The Interrupt Status Register contains flag bits that record an event on the external IRQ pins. When an event as defined in IRQ Rising-Edge Event Enable Register (SIUL2_IREER0) and IRQ Falling-Edge Event Enable Register (SIUL2_IFEER0) occurs, the corresponding flag bit is set. The IRQ Flag bit is set regardless of the state of the corresponding Interrupt Request Enable bit in Interrupt Request Enable Register (SIUL2_IRER0) - Standard. The IRQ Flag bit remains set until cleared by software. The IRQ Flag bits are cleared by writing a '1' to the bits. A write of '0' has no effect. The number of bits implemented is specified by the parameter that defines the number external IRQ pins.

This register holds the interrupt flags and supports 8-bit, 16-bit, and 32-bit accesses.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EIF31	0							EIF23	EIF22	EIF21	EIF20	EIF19	EIF18	EIF17	EIF16
W	w1c								w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIF15	EIF14	EIF13	EIF12	EIF11	EIF10	EIF9	EIF8	EIF7	EIF6	EIF5	EIF4	EIF3	EIF2	EIF1	EIF0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_ISR0 field descriptions

Field	Description
0 EIF31	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 EIF23	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
9 EIF22	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
10 EIF21	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
11 EIF20	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
12 EIF19	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
13 EIF18	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
14 EIF17	External Interrupt Status Flag x

Table continues on the next page...

SIUL2_ISR0 field descriptions (continued)

Field	Description
	This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
15 EIF16	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
16 EIF15	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
17 EIF14	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
18 EIF13	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
19 EIF12	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
20 EIF11	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
21 EIF10	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred

Table continues on the next page...

SIUL2_ISR0 field descriptions (continued)

Field	Description
22 EIF9	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
23 EIF8	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
24 EIF7	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
25 EIF6	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_IRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
26 EIF5	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
27 EIF4	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
28 EIF3	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
29 EIF2	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request.

Table continues on the next page...

SIUL2_ISR0 field descriptions (continued)

Field	Description
	0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
30 EIF1	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred
31 EIF0	External Interrupt Status Flag x This flag can be cleared only by writing 1. Writing 0 has no effect. If enabled (SIUL2_DIRER[x]), EIF[x] causes an interrupt request. 0 No interrupt event has occurred on the pad 1 An interrupt event as defined by SIUL2_IREER[x] and SIUL2_IFEER[x] has occurred

15.2.4 SIUL2 Interrupt Request Enable Register0 (SIUL2_IRER0)

The Interrupt Request Enable Register enables the interrupt request if the corresponding External IRQ Flag bit is set in Interrupt Status Flag Register0. The type of request enabled is determined by the corresponding Interrupt Request Select bit in Interrupt Request Select Register0. The number of bits implemented is specified by the parameter that defines the number of external IRQ pins. This register is used to enable the interrupt messaging to the interrupt controller.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EIRE31	0							EIRE23	EIRE22	EIRE21	EIRE20	EIRE19	EIRE18	EIRE17	EIRE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIRE15	EIRE14	EIRE13	EIRE12	EIRE11	EIRE10	EIRE9	EIRE8	EIRE7	EIRE6	EIRE5	EIRE4	EIRE3	EIRE2	EIRE1	EIRE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IRER0 field descriptions

Field	Description
0 EIRE31	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 EIRE23	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
9 EIRE22	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
10 EIRE21	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
11 EIRE20	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
12 EIRE19	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
13 EIRE18	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
14 EIRE17	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
15 EIRE16	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
16 EIRE15	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
17 EIRE14	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
18 EIRE13	External Interrupt Request Enable x

Table continues on the next page...

SIUL2_IRER0 field descriptions (continued)

Field	Description
	0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
19 EIRE12	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
20 EIRE11	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
21 EIRE10	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
22 EIRE9	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
23 EIRE8	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
24 EIRE7	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
25 EIRE6	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
26 EIRE5	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
27 EIRE4	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
28 EIRE3	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
29 EIRE2	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
30 EIRE1	External Interrupt Request Enable x

Table continues on the next page...

SIUL2_IRER0 field descriptions (continued)

Field	Description
	0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register
31 EIRE0	External Interrupt Request Enable x 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 Set EIF[x] bit causes an interrupt request depending on IRSR register

15.2.5 SIUL2 Interrupt Request Select Register0 (SIUL2_IRSR0)

The IRSR selects the interrupt request. If the corresponding bits are set in Interrupt Status Flag Register0 and Interrupt Request Enable Register0, then the Interrupt Request Select bit determines a interrupt request is asserted.

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DIRSR31	0							DIRSR23	DIRSR22	DIRSR21	DIRSR20	DIRSR19	DIRSR18	DIRSR17	DIRSR16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DIRSR15	DIRSR14	DIRSR13	DIRSR12	DIRSR11	DIRSR10	DIRSR9	DIRSR8	DIRSR7	DIRSR6	DIRSR5	DIRSR4	DIRSR3	DIRSR2	DIRSR1	DIRSR0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IRSR0 field descriptions

Field	Description
0 DIRSR31	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 DIRSR23	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin.

Table continues on the next page...

SIUL2_IRSR0 field descriptions (continued)

Field	Description
	0 Interrupt request is selected 1 Disabled
9 DIRSR22	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
10 DIRSR21	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
11 DIRSR20	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
12 DIRSR19	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
13 DIRSR18	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
14 DIRSR17	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
15 DIRSR16	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
16 DIRSR15	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
17 DIRSR14	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
18 DIRSR13	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin.

Table continues on the next page...

SIUL2_IRSR0 field descriptions (continued)

Field	Description
	0 Interrupt request is selected 1 Disabled
19 DIRSR12	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
20 DIRSR11	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
21 DIRSR10	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
22 DIRSR9	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
23 DIRSR8	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
24 DIRSR7	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
25 DIRSR6	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
26 DIRSR5	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
27 DIRSR4	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
28 DIRSR3	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin.

Table continues on the next page...

SIUL2_IRSR0 field descriptions (continued)

Field	Description
	0 Interrupt request is selected 1 Disabled
29 DIRSR2	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
30 DIRSR1	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled
31 DIRSR0	Interrupt Request Select Register-Selects external interrupt request when an edge-triggered event occurs on the corresponding pin. 0 Interrupt request is selected 1 Disabled

15.2.6 SIUL2 Interrupt Rising-Edge Event Enable Register 0 (SIUL2_IREER0)

This register is used to enable the rising-edge triggered events on the corresponding interrupt pads.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IREE31	0							IREE23	IREE22	IREE21	IREE20	IREE19	IREE18	IREE17	IREE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IREE15	IREE14	IREE13	IREE12	IREE11	IREE10	IREE9	IREE8	IREE7	IREE6	IREE5	IREE4	IREE3	IREE2	IREE1	IREE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IREER0 field descriptions

Field	Description
0 IREE31	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 IREE23	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
9 IREE22	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
10 IREE21	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
11 IREE20	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
12 IREE19	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
13 IREE18	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
14 IREE17	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
15 IREE16	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
16 IREE15	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
17 IREE14	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
18 IREE13	Enable rising-edge events to cause the EIF[x] bit to be set.

Table continues on the next page...

SIUL2_IREER0 field descriptions (continued)

Field	Description
	0 Rising-edge event is disabled 1 Rising-edge event is enabled
19 IREE12	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
20 IREE11	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
21 IREE10	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
22 IREE9	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
23 IREE8	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
24 IREE7	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
25 IREE6	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
26 IREE5	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
27 IREE4	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
28 IREE3	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
29 IREE2	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
30 IREE1	Enable rising-edge events to cause the EIF[x] bit to be set.

Table continues on the next page...

SIUL2_IREER0 field descriptions (continued)

Field	Description
	0 Rising-edge event is disabled 1 Rising-edge event is enabled
31 IREE0	Enable rising-edge events to cause the EIF[x] bit to be set. 0 Rising-edge event is disabled 1 Rising-edge event is enabled

15.2.7 SIUL2 Interrupt Falling-Edge Event Enable Register 0 (SIUL2_IFEER0)

This register is used to enable falling-edge triggered events on the corresponding interrupt pads.

NOTE

If both IREE bit and IFEE bit is cleared for the same interrupt source, the interrupt status flag for the corresponding external interrupt will never be set.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IFEE31	0							IFEE23	IFEE22	IFEE21	IFEE20	IFEE19	IFEE18	IFEE17	IFEE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IFEE15	IFEE14	IFEE13	IFEE12	IFEE11	IFEE10	IFEE9	IFEE8	IFEE7	IFEE6	IFEE5	IFEE4	IFEE3	IFEE2	IFEE1	IFEE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFEER0 field descriptions

Field	Description
0 IFEE31	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIUL2_IFEER0 field descriptions (continued)

Field	Description
8 IFEE23	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
9 IFEE22	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
10 IFEE21	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
11 IFEE20	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
12 IFEE19	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
13 IFEE18	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
14 IFEE17	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
15 IFEE16	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
16 IFEE15	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
17 IFEE14	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
18 IFEE13	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
19 IFEE12	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

Table continues on the next page...

SIUL2_IFEER0 field descriptions (continued)

Field	Description
20 IFEE11	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
21 IFEE10	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
22 IFEE9	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
23 IFEE8	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
24 IFEE7	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
25 IFEE6	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
26 IFEE5	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
27 IFEE4	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
28 IFEE3	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
29 IFEE2	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
30 IFEE1	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled
31 IFEE0	Enable falling-edge events to cause the EIF[x] bit to be set. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

15.2.8 SIUL2 Interrupt Filter Enable Register 0 (SIUL2_IFER0)

This register is used to enable a digital filter counter on the corresponding interrupt pads to filter out glitches on the inputs. The number of interrupts supporting this feature is MCU dependent and can be configured between 0 and 31.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	IFE31	0							IFE23	IFE22	IFE21	IFE20	IFE19	IFE18	IFE17	IFE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	IFE15	IFE14	IFE13	IFE12	IFE11	IFE10	IFE9	IFE8	IFE7	IFE6	IFE5	IFE4	IFE3	IFE2	IFE1	IFE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFER0 field descriptions

Field	Description
0 IFE31	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 IFE23	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
9 IFE22	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
10 IFE21	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
11 IFE20	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

Table continues on the next page...

SIUL2_IFER0 field descriptions (continued)

Field	Description
12 IFE19	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
13 IFE18	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
14 IFE17	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
15 IFE16	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
16 IFE15	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
17 IFE14	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
18 IFE13	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
19 IFE12	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
20 IFE11	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
21 IFE10	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
22 IFE9	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
23 IFE8	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

Table continues on the next page...

SIUL2_IFER0 field descriptions (continued)

Field	Description
24 IFE7	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
25 IFE6	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
26 IFE5	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
27 IFE4	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
28 IFE3	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
29 IFE2	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
30 IFE1	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled
31 IFE0	Enable digital glitch filter on the interrupt pad input. 0 Filter is disabled 1 Filter is enabled

15.2.9 SIUL2 Interrupt Filter Maximum Counter Register (SIUL2_IFMCR*n*)

These registers are used to configure the filter counter associated with each digital glitch filter.

Address: 0h base + 40h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												MAXCNT			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFMCR*n* field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 MAXCNT	Maximum Interrupt Filter Counter setting Filter Period = TCK*MAXCNTx + n*TCK where (n can be 0 to 4) MAXCNTx can be 0 to 15 (for MAXCNT<3 filter will behave as all PASS filter) TCK Prescaled Filter Clock Period, which is IRC clock prescaled to IFCP value T(IRC) Basic Filter Clock Period: 62.5 ns (F=16 MHz) Also note that Filter delay is 2 TCK clock cycles more than Filter period

15.2.10 SIUL2 Interrupt Filter Clock Prescaler Register (SIUL2_IFCPR)

This register is used to configure a clock prescaler which is used to select the clock for all digital filter. Prescaler is applied to the input clock to the SIUL2, which is the system AIPS clock counters in the SIUL2.

NOTE

The glitch filters need a running internal oscillator clock to work. If no such clock is available, external interrupts will be effectively disabled when the glitch filter is enabled on a interrupt line.

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												IFCP			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_IFCPR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 IFCP	Interrupt Filter Clock Prescaler setting Prescaled Filter Clock Period = T(IRC) x (IFCP + 1) T(IRC) is the internal oscillator period. IFCP can be 0 to 15

15.2.11 SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR*n*)

Selects which source signal is connected to the register's associated destination, which is a chip pin that is or can be configured as an output.

For the associated chip-pin destination, the register also specifies the pin's electrical properties.

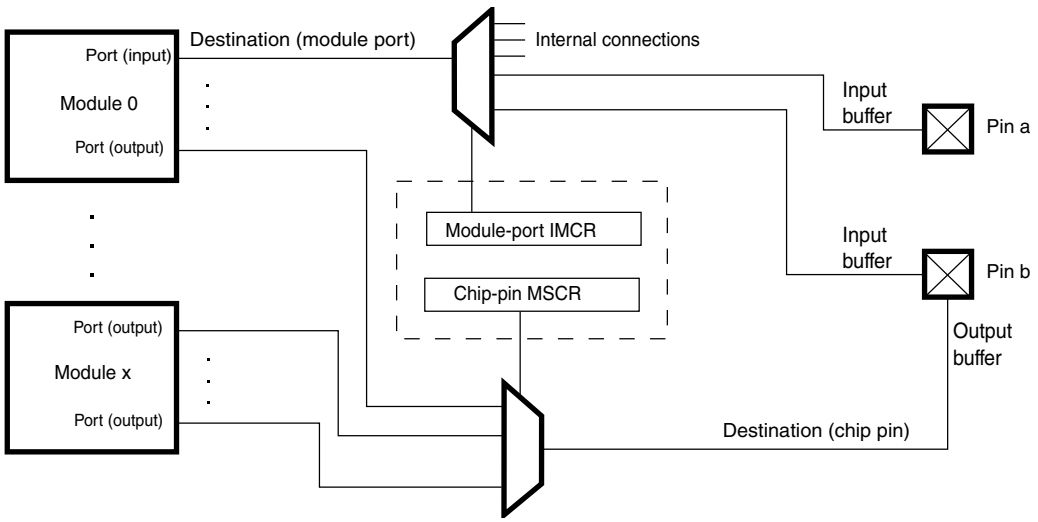


Figure 15-2. MSCR pin connection

The fields in an MSCR vary depending on its associated destination (chip pin).

For available chip-pin MSCR assignments, pin types, bit availability on each MSCR, APC support, and SSS values etc. see the I/O Signal Description and Input Multiplexing Tables (excel file) attached to this document.

This register only supports 32-bit access. Byte and half-word write accesses are not supported.

Address: 0h base + 240h offset + (4d × i), where i=0d to 263d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		SRC[1:0]		0		OBE	ODE	SMC	APC	0	0	IBE	HYS	PUS	PUE
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	1*	0*	0*	0*	0*	1*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								0				SSS			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- See the Signal Description details for the reset values and the availability of pad setting bits in MSCRs.

SIUL2_MSCR*n* field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–3 SRC[1:0]	Slew Rate Control Used only when the associated destination is a chip pin. 00 Half drive strength with slew rate control 01 Full drive strength with slew rate control 10 Half drive strength without slew rate control 11 Full drive strength without slew rate control
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 OBE	GPIO Output Buffer Enable Applies only to digital pins. Otherwise this bit is reserved. 0 Output driver disabled 1 Output driver enabled
7 ODE	Open Drain Enable 0 Open drain function disabled 1 Open drain function enabled, when OBE is also 1.
8 SMC	Safe Mode Control Used only when the associated destination is a chip pin. Specifies whether the chip disables the pin's output buffer when the chip enters Safe mode. 0 Disable (the output buffer returns to its previous state when the chip leaves Safe mode) 1 Don't disable
9 APC	Analog Pad Control Used only when the associated destination is a chip pin that supports analog I/O. Enables the pin's analog-input-path switch. 0 Disable (the switch is off) 1 Enable (another module can control the state of the switch)
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 IBE	Input Buffer Enable Used only when the associated destination is a chip pin. Enables the associated pin's input buffer. 0 Disabled 1 Enabled
13 HYS	Input Hysteresis Used only when the associated destination is a chip pin. Enables input hysteresis for the associated pin.

Table continues on the next page...

SIUL2_MSCR*n* field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
14 PUS	Pull Select Determines whether the pull function is a pullup or pulldown when the pull function is enabled by the PUE field. Used only when the associated destination is a chip pin. 0 Pulldown 1 Pullup
15 PUE	Pull Enable Enables the pull function. Used only when the associated destination is a chip pin. 0 Disabled 1 Enabled
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (chip pin or module port). This field has a variable width depending on the number of alternate functions available for that pin/port. For a chip pin, the source signals are outputs from module ports. For a module port, the source signals are either outputs from module ports or inputs from chip pins. See the device I/O Signal Description and Input Multiplexing Tables (excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the excel file to open it and select the I/O Signal Description Table tab.

15.2.12 SIUL2 Input Multiplexed Signal Configuration Register (SIUL2_IMCR*n*)

Selects which source signal is connected to the register's associated destination, which is an internal module port that is or can be configured as an input.

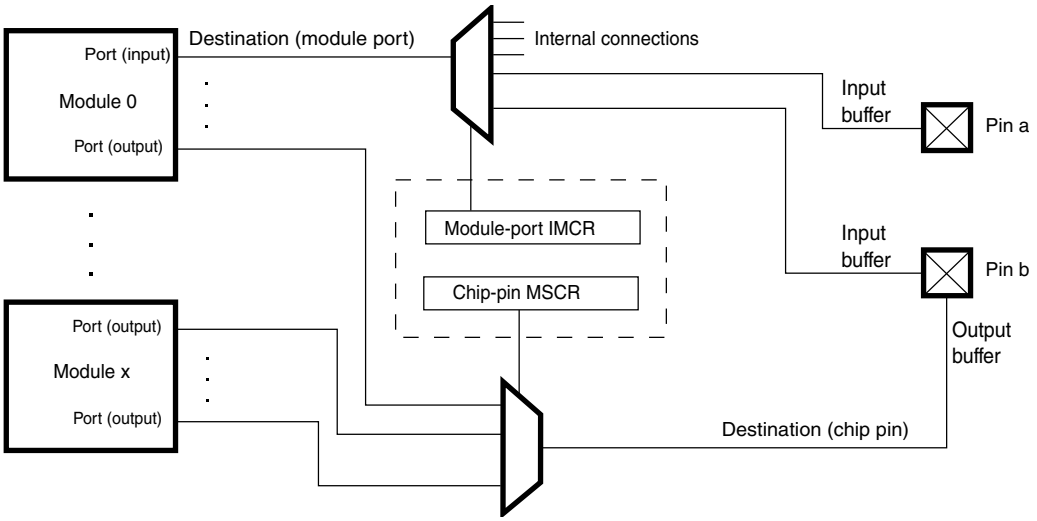


Figure 15-3. IMCR module-port connection(For IMCR0 to IMCR27, IMCR32 to IMCR63, IMCR68 to IMCR511)

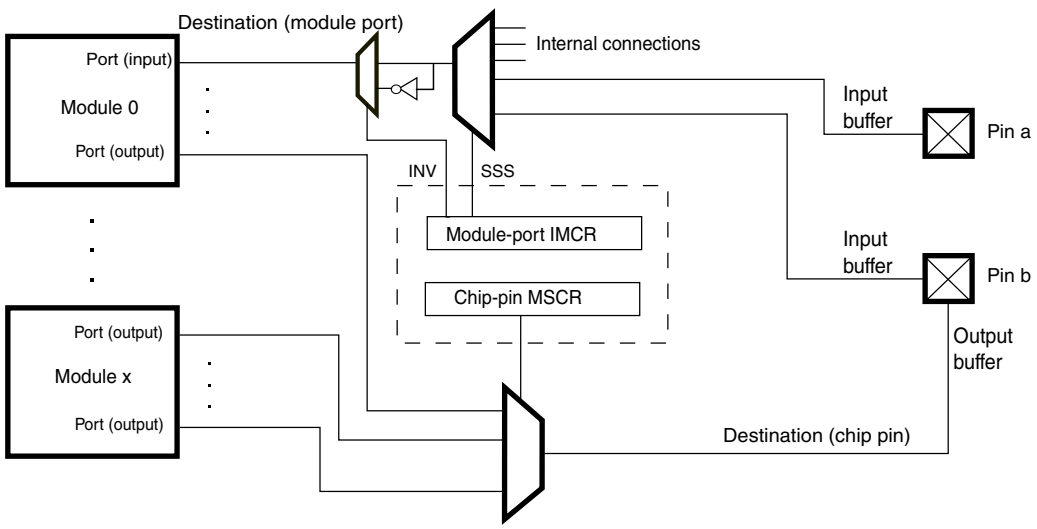


Figure 15-4. IMCR module-port connection (For IMCR28 to IMCR31 and IMCR64 to IMCR67)

The fields in an IMCR vary depending on its associated destination (module port).

For IMCR assignments and SSS values, see the I/O Signal Description and Input Multiplexing Tables (excel file) attached to this document.

Address: 0h base + A40h offset + (4d × i), where i=0d to 511d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INV	0														
W																
Reset	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								0				SSS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:

- INV field: INV bit exist only for IMCR28, IMCR29, IMCR30, IMCR31, IMCR64, IMCR65, IMCR66, and IMCR67. For rest of the IMCRs this field is reserved.

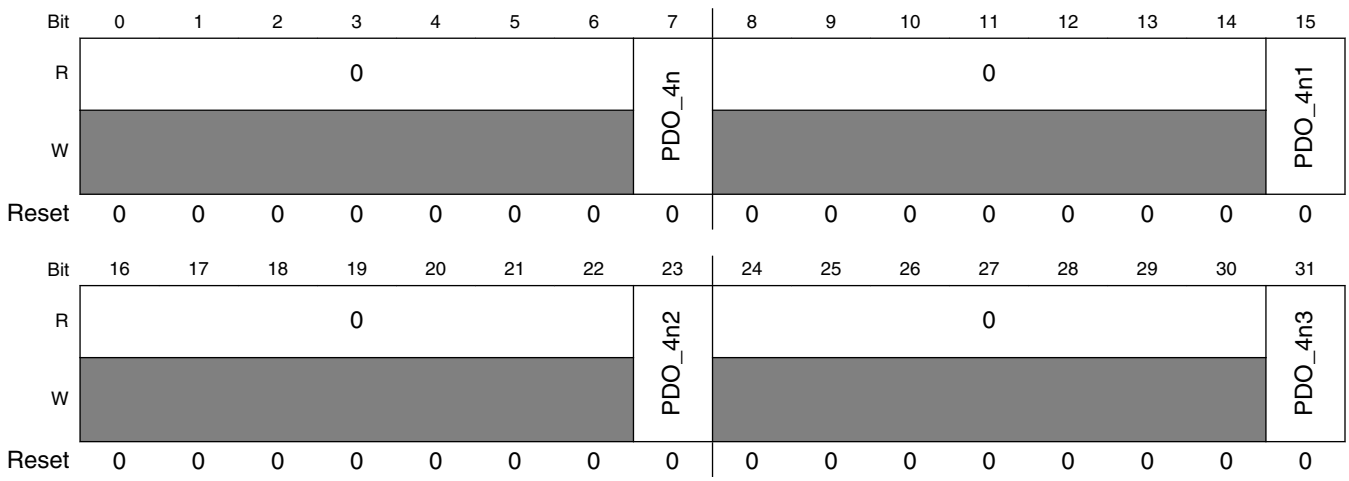
SIUL2_IMCR*n* field descriptions

Field	Description
0 INV	Invert This bit is used to invert the source signal. 0 Source signal is connected to the associated destination directly either from chip pin or module port. 1 Source signal is inverted before it is connected to the associated destination directly either from chip pin or module port.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 SSS	Source Signal Select Selects which source signal is connected to the associated destination (chip pin or module port). This field has a variable width depending on the number of alternate functions available for that pin/port. For a chip pin, the source signals are outputs from module ports. For a module port, the source signals are either outputs from module ports or inputs from chip pins. See the device I/O Signal Description and Input Multiplexing Tables (excel file) attached to this document. Locate the paperclip symbol on the left side of the PDF window, and click it. Double-click on the excel file to open it and select the I/O Signal Description Table tab.

15.2.13 SIUL2 GPIO Pad Data Output Register (SIUL2_GPDO*n*)

These registers can be used to set or clear a single GPIO pad with a byte access.

Address: 0h base + 1300h offset + (4d × i), where i=0d to 65d



SIUL2_GPDO*n* field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PDO_4n	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. PDO_4n represents PDO[4n], where n is the instance of the register. For example, for the GPDO3 register, PDO_4n represents PDO12. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PDO_4n1	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. PDO_4n1 represents PDO[4n+1], where n is the instance of the register. For example, for the GPDO3 register, PDO_4n1 represents PDO13. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 PDO_4n2	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register.

Table continues on the next page...

SIUL2_GPDO*n* field descriptions (continued)

Field	Description
	PDO_4n2 represents PDO[4n+2], where n is the instance of the register. For example, for the GPDO3 register, PDO_4n2 represents PDO14. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PDO_4n3	Pad Data Out This bit stores the data to be driven out on the external GPIO pad controlled by this register. PDO_4n3 represents PDO[4n+3], where n is the instance of the register. For example, for the GPDO3 register, PDO_4n3 represents PDO15. 0 Logic low value is driven on the corresponding GPIO pad when the pad is configured as an output 1 Logic high value is driven on the corresponding GPIO pad when the pad is configured as an output

15.2.14 SIUL2 GPIO Pad Data Input Register (SIUL2_GPDI*n*)

These registers can be used to read the GPIO pad data with a byte access.

Address: 0h base + 1500h offset + (4d × i), where i=0d to 65d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							PDI_4n	0							PDI_4n1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							PDI_4n2	0							PDI_4n3
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_GPDI*n* field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 PDI_4n	Pad Data In This bit stores the value of the external GPIO pad associated with this register. PDI_4n represents PDI[4n], where n is the instance of the register. For example, for the GPDI3 register, PDI_4n represents PDI12. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PDI_4n1	Pad Data In This bit stores the value of the external GPIO pad associated with this register. PDI_4n1 represents PDI[4n+1], where n is the instance of the register. For example, for the GPDI3 register, PDI_4n1 represents PDI13. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 PDI_4n2	Pad Data In This bit stores the value of the external GPIO pad associated with this register. PDI_4n2 represents PDI[4n+2], where n is the instance of the register. For example, for the GPDI3 register, PDI_4n2 represents PDI14. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PDI_4n3	Pad Data In This bit stores the value of the external GPIO pad associated with this register. PDI_4n3 represents PDI[4n+3], where n is the instance of the register. For example, for the GPDI3 register, PDI_4n3 represents PDI15. 0 The value of the data in signal for the corresponding GPIO pad is logic low 1 The value of the data in signal for the corresponding GPIO pad is logic high

15.2.15 SIUL2 Parallel GPIO Pad Data Out Register (SIUL2_PGPDO*n*)

These registers are used to set or clear the respective pads of the device. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration. The difference between PGPDO and GPDO is that PGPDO registers can be used to set the values of all output pins assigned to a device port with a single 16-bit register write vs. the GPDO registers, which are used to set the value on a specific pin with a byte write.

The SIUL2_PGPDO registers access the same physical resource as the SIUL2_PDO and SIUL2_MPGPDO address locations. Some examples of the mapping:

- PPDO[0][0] = PDO[0]
- PPDO[2][0] = PDO[32]
- PPDO[31][15] = PDO[511]

Address: 0h base + 1700h offset + (2d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	PPDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_PGPDO*n* field descriptions

Field	Description
0–15 PPDO	Parallel Pad Data Out Write or read the data register that stores the value to be driven on the pad in output mode. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Output Registers (SIUL2_GPDO).The x and bit index define which PPDO register bit is equivalent to which PDO register bit according to the following equation: PPDO[x][y] = PDO[(x*16)+y]

15.2.16 SIUL2 Parallel GPIO Pad Data In Register (SIUL2_PGPDI*n*)

These registers hold the synchronized input value from the pads. Parallel port registers for input (PGPDI) and output (PGPDO) are provided to allow a complete port to be written or read in one operation, dependent on the individual pad configuration.The difference between PGPDI and GPDI registers is that PGPDI registers can be used to read the values of all input pins assigned to a device port with a single 16-bit register read vs. the GPDI registers, which are used to read the value on a specific pin with a byte read.

Address: 0h base + 1740h offset + (2d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	PPDI															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_PGPDI*n* field descriptions

Field	Description
0–15 PPDI	Parallel Pad Data In Reads the current pad value. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Input Registers (SIUL2_GPDI). The x and bit index define which PPDI register bit is equivalent to which PDI register bit according to the following equation: PPDI[x][y] = PDI[(x*16)+y]

15.2.17 SIUL2 Masked Parallel GPIO Pad Data Out Register (SIUL2_MPGPDO*n*)

This register can be used to selectively modify the pad values associated to PPDO[x] [0:15]. The MPGPDO[x] register may only be accessed with 32-bit writes. 8-bit or 16-bit writes will not modify any bits in the register and cause a transfer error response by the module. Read access will return 0.

Address: 0h base + 1780h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																0															
W	MASK																MPPDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SIUL2_MPGPDO*n* field descriptions

Field	Description
0–15 MASK	Mask Field Each bit corresponds to one data bit in the MPPDO[x] register at the same bit location. 0b The associated bit value in the MPPDO[x] field is ignored 1b The associated bit value in the MPPDO[x] field is written
16–31 MPPDO	Masked Parallel Pad Data Out Write the data register that stores the value to be driven on the pad in output mode. Access to this register location are coherent with access to the bit-wise GPIO Pad Data Output Registers (PDO).The x and bit index define which MPPDO register bit is equivalent to which PDO register bit according to the following equation: MPPDO[x][y] = PDO[(x*16)+y]

15.3 Functional description

15.3.1 General

This section provides a complete functional description of the System Integration Unit Lite2.

15.3.2 Pad Control

The SIUL2 is capable of controlling the electrical characteristic of up to 264 pads. It provides a consistent interface for all pads, both on a by-port and a by-bit basis. The following describes the pad characteristics that can be supported by the SIUL2. Not all of these features will be supported on all devices or on all pads as this is dependent on the specific application needs an the pads implemented on the device.

The required controls are:

- Slew rate control
  - Offers improved EMC performance.
  - Support for a fast and a slow slew rate.
- Drive strength control

    - Offers improved EMC performance.
    - Up to 4 different drive strength levels can be supported.
- Pull capability
    - Offers flexibility in the device configuration and the elimination of external hardware in some cases.
    - The configuration of the pull on any pad should be independently controlled to be either pull-up, pull-down or no-pull enabled.
- Input and Out Buffer control
- Hysteresis
- Open drain
    - Needed to support different pads muxed (e.g. mux I2C, with non open drain pads)
- Pad Output Assignment
    - This setting defines which chip function has control over the output of the pad.

The setting of each pad out of reset is fixed per MCU, but can be configured individually. In this way it is possible to select special pull settings or peripheral pad ownership per design.

It is possible for the user software to configure each pad independently of all other pads on the device or other pads grouped within a single port. This allows different pad types to be grouped together in ports and allows the necessary flexibility needed for the pads individual operation. This is achieved by grouping all of the above functions into a single register for each pad on the device, and allows each pad to be configured with a single write to one register and allowing simplified duplication of software for each pad with indexed changes for each pad.

15.3.3 General Purpose Input and Output Pads (GPIO)

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

GPIO pads can also optionally be implemented without any alternate function.

The SIUL2 is designed to manage up to 264 GPIO pads organized as ports that can be accessed for data reads and writes as 32-bit.

All port access are identical with each read or write being performed only at a different location to access a different port width.

This requires that the registers are arranged in such a way as to support this range of port widths without having to split reads or writes into multiple accesses.

The module has separate data input and data output registers for all pads, allowing the possibility of reading back an input or output value of a pad directly. This supports the ability to validate what is present on the pad rather than simply confirming the value that was written to the data register by accessing the data input registers.

The data output registers support both read and write operations to be performed.

The data input registers support read access only.

When the pad is configured to use one of its alternate functions, the data input value reflect the respective value of the pad. If a write operation is performed to the data output register for a pad configured as an alternate function (non GPIO), this write will not be reflected by the pad value until re-configured to GPIO. All general purpose pads are implemented as bidirectional.

Note

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

15.3.4 External Interrupts Requests (REQ Pins)

The SIUL2 supports 0 to 31 external interrupts, which can be allocated to any pad necessary at the MCU level. This allocation is fixed per MCU.

For the assignments of the external interrupts, see the Signal Description details.

The SIUL2 supports up to 4 interrupt vectors to the interrupt controller of the MCU. Each interrupt vector can support up to 8 external interrupt sources from the device pads.

For devices with a number of interrupt sources that is smaller than the maximum number, the MCU can implement either all four interrupt vectors with fewer sources or reduce the number of vectors as the number of sources are reduced. The following figure shows two sample scenarios.

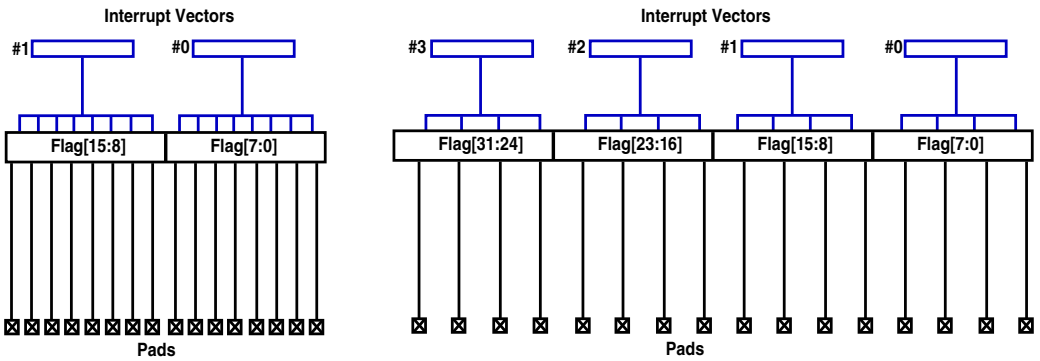


Figure 15-5. Example of different interrupt to vector mapping at MCU level

All the external interrupt pads within a single group (maximum 8 pads) have equal priority. It is the responsibility of the user software to search through the group of sources in the most appropriate way for their application.

The priority of the vectors used by the external interrupt pads is fixed based on the platform and the interrupt controller and its priority levels, but the allocation of pads to each group of interrupts can be independently configured by the MCU.

An MCU-specific number of external interrupt lines can have a digital glitch filters applied to them. The supported range is up to 32.

15.3.4.1 External interrupt initialization

When an external interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request, during pin interrupt initialization, the user must do the following:

1. Mask interrupts by clearing the EIREn bits in IRER0.
2. Select the pin polarity by setting the appropriate IREEn bits in IREER0 and the appropriate IFEEn bits in IFEER0 as desired.
3. Configure the appropriate bits in the MSCR[0:511] register for the external interrupt pin(s) desired as follows:
   - Clear the OBE and ODE bits to disable output.
   - Set the IBE bit to enable the pin's input buffer.
   - If using the internal pullup/pulldown, configure the appropriate PUE and PUS fields.

Note

MSCR[SSS] bits do not need to be changed for either MSCR[0:511] or IMCR[0:511] because the external interrupt pin input is directly connected to the SIUL for EIRQ pins.

External interrupt pins should never be configured as outputs (MSCR[ODC] bits are not zeros) when external interrupt inputs are desired because false interrupts could be detected (such as from a GPIO configuration).

4. Select the request desired Interrupt by writing the appropriate IRSn bits in IRSR0.
5. Select the desired glitch filter setup for the pins by writing the following:
   - Write the Filter Counter setting to the desired value by writing the MAXCNT[3:0] bits in the IFMCn register for the respective external interrupt that is being used.
   - Set the Filter Clock Prescaler setting from 0 to 15 to the desired value by writing the IFCP[3:0] bits in the IFCPR register.
   - Enable the glitch filter for the desired external interrupt pins by setting the appropriate IFEn bits in IFER0.

NOTE

The glitch filter clock runs on the internal RC oscillator clock whereas the SIUL logic runs off of the PBRIDGEx clock. The IRC clock is at a fixed frequency of 16 MHz. The PBRIDGEx clock can run as high as 40 MHZ when core clock is at 160 MHz. The glitch filter is enabled on IRC clock, so make sure at least one IRC clock occurs after enabling it before trying to detect glitchless events.

6. Write to EIFn bits in ISR0 to as desired to clear any flags.
7. Enable the interrupt pins by setting the appropriate EIREn bits in IRER0.

15.3.4.2 External interrupt management

The user can enable or disable each interrupt independently using a single rolled up register, SIUL2_IRER0.

A pad defined as an external interrupt can be configured by the user to recognize interrupts with an active rising edge, an active falling edge, or both edges being active. A setting of having both edge events disabled is reserved and should not be configured. The user controls the active IRQ edge through the registers SIUL2_IREER and SIUL2_IFEER.

Each external interrupt supports an individual flag which is held in the Flag register (ISR0). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register.

Refer to the following figure for an overview of the external interrupt implementation.

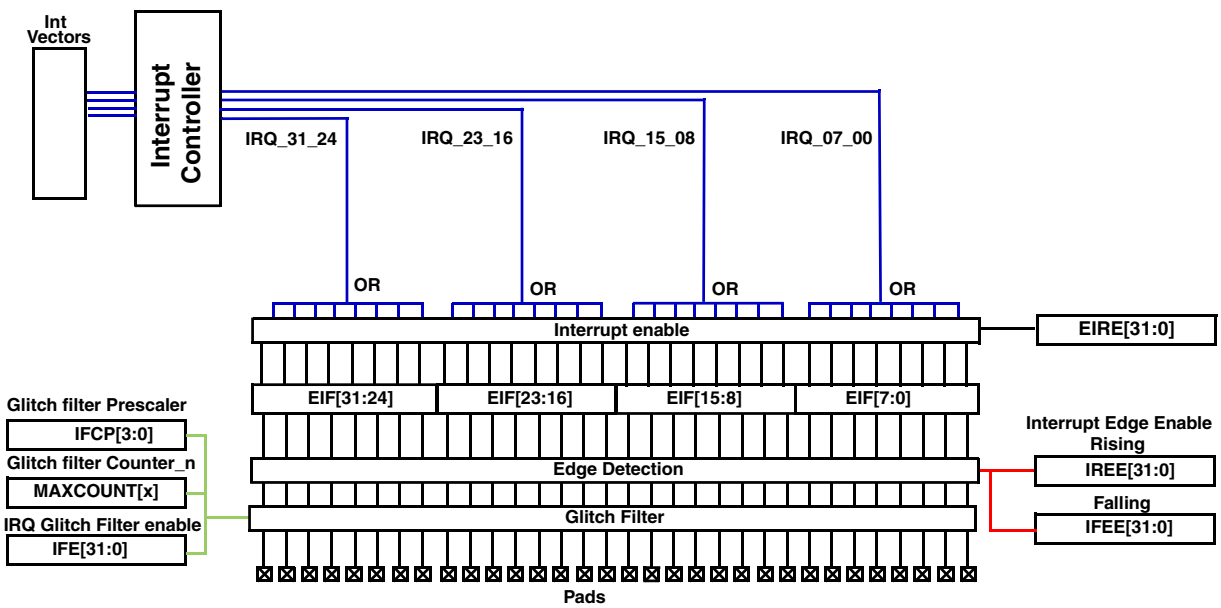


Figure 15-6. External interrupt pad diagram

15.3.4.3 External Interrupt request

The EIRQ input pins on the device are sources for interrupt requests. The system has four possible interrupt vectors available for the EIRQ pins in the SIUL2. The mapping of 32 interrupt request sources to vectors and channels appears in the following table.

Table 15-1. Interrupt source mapping to SIUL2 Interrupt request output for 32 interrupt sources

Vector/Channel #	Interrupt Vector Source
0	EIRQ[07] | EIRQ[06] | EIRQ[05] | EIRQ[04] | EIRQ[03] | EIRQ[02] | EIRQ[01] | EIRQ[00]
1	EIRQ[15] | EIRQ[14] | EIRQ[13] | EIRQ[12] | EIRQ[11] | EIRQ[10] | EIRQ[09] | EIRQ[08]
2	EIRQ[23] | EIRQ[22] | EIRQ[21] | EIRQ[20] | EIRQ[19] | EIRQ[18] | EIRQ[17] | EIRQ[16]
3	EIRQ[31] | EIRQ[30] | EIRQ[29] | EIRQ[28] | EIRQ[27] | EIRQ[26] | EIRQ[25] | EIRQ[24]

Chapter 16
Crossbar Switch (AXBS)

16.1 Chip-specific AXBS information

16.1.1 Crossbar Switch configuration

This section summarizes how the AXBS_0 is configured in the chip.

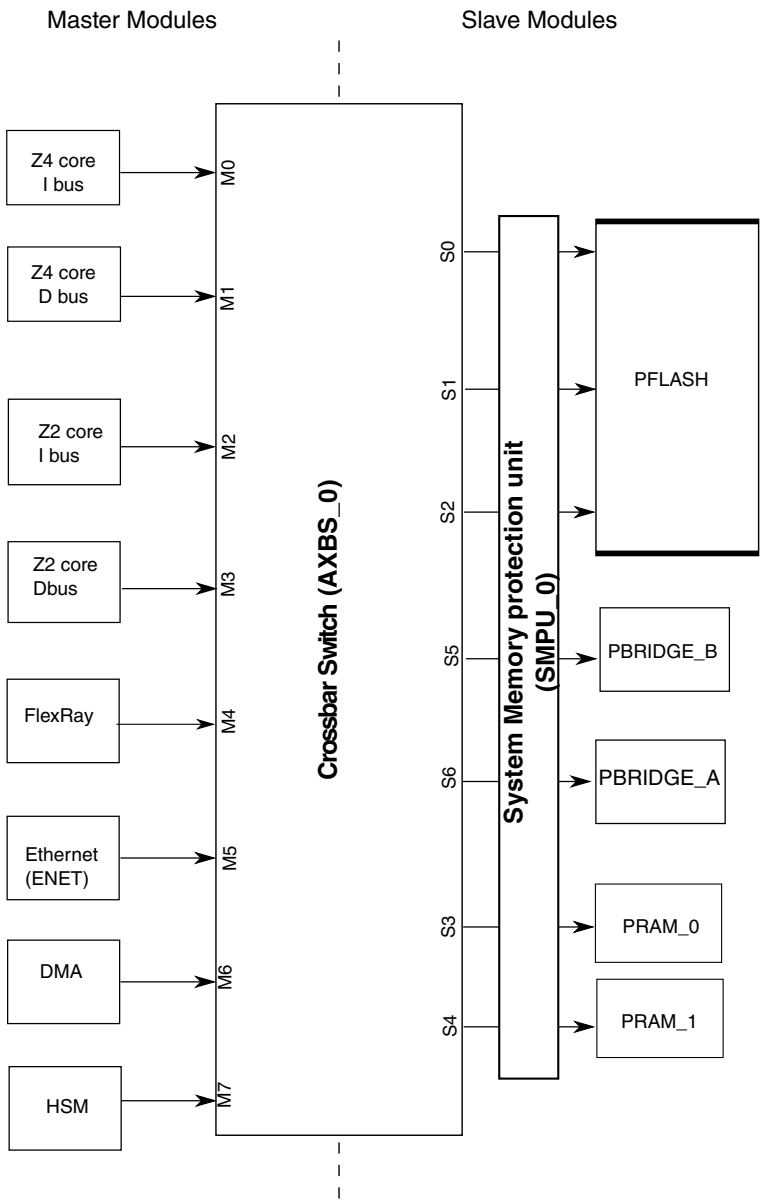


Figure 16-1. AXBS 0 integration

16.1.2 AXBS 0 Master and Slave assignments

This device contains 8 masters connected to the crossbar switch 0 (AXBS_0).

The master and slave assignments are as follows:

Table 16-1. AXBS_0 (Key: x = available connection)

Master Ports	Slave Ports							
Master	Master Port No.	0	1	2	3	4	5	6
		(PFLASH__ 0)	(PFLASH_1)	(PFLASH_2)	(PBRIDGE_ A)	(PBRIDGE_ B)	PRAM_0	PRAM_1
Z4 I bus	0	**x**[1]			**x**	x	x	x
Z4 D bus	1		x		x	x	x[2]	x
FlexRay	2						x	x[3]
ENET	3						x	x
Z2 I bus	4		x[4]		x	x	x	x
Z2 D bus	5	x			x[5]	x[6]	x	x
DMA	6			x	x	x	x	x
HSM	7			x[7]	x	x	x	x

1. Slave Port No. 0 should be set to “park on last”. Arbitration mode should be set to fixed priority and Master Port No. 0 (Z4_I) should be given highest priority when accessing Slave Port No. 0 by default.
2. Slave Port No. 5 should be set to “park on last”. Arbitration mode should be set to fixed priority and Master Port No. 1(Z4_D) should be given highest priority when accessing Slave Port No. 5 by default
3. Slave Port No. 6 should be set to “park on last”. Arbitration mode should be set to fixed priority and Master Port No. 2(FlexRay) should be given highest priority when accessing Slave Port No. 6 by default
4. Slave Port No. 1 should be set to “park on last”. Arbitration mode should be set to fixed priority and Master Port No. 4(Z2_I) should be given highest priority when accessing Slave Port No. 1 by default.
5. Slave Port No. 3 should be set to “park on last”. Arbitration mode should be set to fixed priority and Master Port No. 5(Z2_D) should be given highest priority when accessing Slave Port No. 3 by default
6. Slave Port No. 4 should be set to “park on last”. Arbitration mode should be set to fixed priority and Master Port No. 5(Z2_D) should be given highest priority when accessing Slave Port No. 4 by default
7. Slave Port No. 2 should be set to “park on last”. Arbitration mode should be set to fixed priority and Master Port No. 7(HSM) should be given highest priority when accessing Slave Port No. 2 by default.

16.1.3 PRS register reset values

The AXBS0_PRS*n* registers reset to 7654_3210h.

16.1.4 CRS register reset values

The AXBS0_CRS*n* registers reset to 00FF_0010h.

NOTE

To enable DMA master HPE mode, DMA_TCDn_CSR[BWC] field must be set to 0x01.

16.2 Introduction

The information found here provides information on the layout, configuration, and programming of the crossbar switch.

The crossbar switch connects bus masters and bus slaves using a crossbar switch structure. This structure allows all bus masters to access different bus slaves simultaneously, while providing arbitration among the bus masters when they access the same slave. A variety of bus arbitration methods and attributes may be programmed on a slave-by-slave basis.

16.2.1 Features

The crossbar switch includes these features:

- Symmetric crossbar bus switch implementation
  - Allows concurrent accesses from different masters to different slaves
  - Slave arbitration attributes configured on a slave-by-slave basis
- Up to single-clock 64-bit transfer
- Support for burst transfers of up to 16 beats of data
- Low-Power Park mode support
- Dynamic master priority elevation

16.3 Memory Map / Register Definition

Each slave port of the crossbar switch contains configuration registers. Read- and write-transfers require two bus clock cycles. The registers can be read from and written to only in supervisor mode. Additionally, these registers can be read from or written to only by 32-bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The CRS*n* and PRS*n* registers can be programmed to be read-only to prevent changes to their configuration. After being read-only protected, future writes to them will terminate with an error.

NOTE

This section shows the registers for all eight master and slave ports. If a master or slave is not used on this particular device, then unexpected results occur when writing to its registers. See the chip configuration details for the exact master/slave assignments for your device.

All references to the crossbar switch registers are based on the physical port connections.

AXBS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Priority Registers Slave (AXBS_PRS0)	32	R/W	See section	16.3.1/382
10	Control Register (AXBS_CRS0)	32	R/W	See section	16.3.2/385
100	Priority Registers Slave (AXBS_PRS1)	32	R/W	See section	16.3.1/382
110	Control Register (AXBS_CRS1)	32	R/W	See section	16.3.2/385
200	Priority Registers Slave (AXBS_PRS2)	32	R/W	See section	16.3.1/382
210	Control Register (AXBS_CRS2)	32	R/W	See section	16.3.2/385
300	Priority Registers Slave (AXBS_PRS3)	32	R/W	See section	16.3.1/382
310	Control Register (AXBS_CRS3)	32	R/W	See section	16.3.2/385
400	Priority Registers Slave (AXBS_PRS4)	32	R/W	See section	16.3.1/382
410	Control Register (AXBS_CRS4)	32	R/W	See section	16.3.2/385
500	Priority Registers Slave (AXBS_PRS5)	32	R/W	See section	16.3.1/382
510	Control Register (AXBS_CRS5)	32	R/W	See section	16.3.2/385
600	Priority Registers Slave (AXBS_PRS6)	32	R/W	See section	16.3.1/382
610	Control Register (AXBS_CRS6)	32	R/W	See section	16.3.2/385
700	Priority Registers Slave (AXBS_PRS7)	32	R/W	See section	16.3.1/382
710	Control Register (AXBS_CRS7)	32	R/W	See section	16.3.2/385
800	Master General Purpose Control Register (AXBS_MGPCR0)	32	R/W	0000_0000h	16.3.3/387
900	Master General Purpose Control Register (AXBS_MGPCR1)	32	R/W	0000_0000h	16.3.3/387
A00	Master General Purpose Control Register (AXBS_MGPCR2)	32	R/W	0000_0000h	16.3.3/387
B00	Master General Purpose Control Register (AXBS_MGPCR3)	32	R/W	0000_0000h	16.3.3/387
C00	Master General Purpose Control Register (AXBS_MGPCR4)	32	R/W	0000_0000h	16.3.3/387
D00	Master General Purpose Control Register (AXBS_MGPCR5)	32	R/W	0000_0000h	16.3.3/387
E00	Master General Purpose Control Register (AXBS_MGPCR6)	32	R/W	0000_0000h	16.3.3/387
F00	Master General Purpose Control Register (AXBS_MGPCR7)	32	R/W	0000_0000h	16.3.3/387

16.3.1 Priority Registers Slave (AXBS_PRS*n*)

The priority registers (PRSn) set the priority of each master port on a per slave port basis and reside in each slave port. The priority register can be accessed only with 32-bit accesses. After the CRSn[RO] bit is set, the PRSn register can only be read; attempts to write to it have no effect on PRSn and result in a bus-error response to the master initiating the write.

Two available masters must not be programmed with the same priority level. Attempts to program two or more masters with the same priority level result in a bus-error response and the PRSn is not updated.

NOTE

Valid values for the M*n* priority fields depend on which masters are available on the chip. This information can be found in the chip-specific information for the crossbar.

- If the chip contains less than five masters, values 0 to 3 are valid. Writing other values will result in an error.
- If the chip contains five or more masters, valid values are 0 to n-1, where n is the number of masters attached to the Crossbar Switch. Other values will result in an error.

NOTE

See the chip-specific crossbar information for the reset value of this register.

Address: 0h base + 0h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	M7			0	M6			0	M5			0	M4		
W																
Reset	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	M3			0	M2			0	M1			0	M0		
W																
Reset	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*	0	0*	0*	0*

* Notes:
- See the chip-specific crossbar information for the reset value of this register.

AXBS_PRS*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

AXBS_PRS*n* field descriptions (continued)

Field	Description
1–3 M7	Master 7 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 M6	Master 6 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 M5	Master 5 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 M4	Master 4 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

AXBS_PRS*n* field descriptions (continued)

Field	Description
17–19 M3	Master 3 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 M2	Master 2 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 M1	Master 1 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 M0	Master 0 Priority. Sets the arbitration priority for this port on the associated slave port. 000 This master has level 1, or highest, priority when accessing the slave port. 001 This master has level 2 priority when accessing the slave port. 010 This master has level 3 priority when accessing the slave port. 011 This master has level 4 priority when accessing the slave port. 100 This master has level 5 priority when accessing the slave port. 101 This master has level 6 priority when accessing the slave port. 110 This master has level 7 priority when accessing the slave port. 111 This master has level 8, or lowest, priority when accessing the slave port.

16.3.2 Control Register (AXBS_CRS*n*)

These registers control several features of each slave port and must be accessed using 32-bit accesses. After CRSn[RO] is set, the PRSn can only be read; attempts to write to it have no effect and result in an error response.

NOTE

See the chip-specific crossbar information for the reset value of this register.

Not all HPE*n* fields may be active. See the chip-specific crossbar information for which masters support high priority elevation. Setting a field corresponding to a master that does not support high-priority elevation has no effect.

Address: 0h base + 10h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RO	HLP	0						HPE7	HPE6	HPE5	HPE4	HPE3	HPE2	HPE1	HPE0
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						ARB		0		PCTL		0	PARK		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0	1	0*	0*	0*	0*

* Notes:
- See the chip-specific crossbar information for the reset value of this register.

AXBS_CRS*n* field descriptions

Field	Description
0 RO	Read Only Forces the PRS*n* and CRS*n* registers to be read-only. After being set, only a hardware reset clears this field. 0 The CRS*n* and PRS*n* registers are writeable 1 The CRS*n* and PRS*n* registers are read-only and cannot be written (attempted writes have no effect on the registers and result in a bus error response).
1 HLP	Halt Low Priority Sets the initial arbitration priority for low power mode requests . Setting this bit will not affect the request for low power mode from attaining highest priority once it has control of the slave ports. 0 The low power mode request has the highest priority for arbitration on this slave port 1 The low power mode request has the lowest initial priority for arbitration on this slave port

Table continues on the next page...

AXBS_CRS*n* field descriptions (continued)

Field	Description
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 HPE7	On this slave port, enable priority elevation for master 7. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 7 is disabled on this slave port 1 Priority elevation for master 7 is enabled on this slave port
9 HPE6	On this slave port, enable priority elevation for master 6. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 6 is disabled on this slave port 1 Priority elevation for master 6 is enabled on this slave port
10 HPE5	On this slave port, enable priority elevation for master 5. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 5 is disabled on this slave port 1 Priority elevation for master 5 is enabled on this slave port
11 HPE4	On this slave port, enable priority elevation for master 4. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 4 is disabled on this slave port 1 Priority elevation for master 4 is enabled on this slave port
12 HPE3	On this slave port, enable priority elevation for master 3. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 3 is disabled on this slave port 1 Priority elevation for master 3 is enabled on this slave port
13 HPE2	On this slave port, enable priority elevation for master 2. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 2 is disabled on this slave port 1 Priority elevation for master 2 is enabled on this slave port
14 HPE1	On this slave port, enable priority elevation for master 1. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 1 is disabled on this slave port 1 Priority elevation for master 1 is enabled on this slave port
15 HPE0	On this slave port, enable priority elevation for master 0. If enabled, the master is able to elevate its priority to the highest. 0 Priority elevation for master 0 is disabled on this slave port 1 Priority elevation for master 0 is enabled on this slave port
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 ARB	Arbitration Mode Selects the arbitration policy for the slave port. 00 Fixed priority 01 Round-robin (rotating) priority

Table continues on the next page...

AXBS_CRS*n* field descriptions (continued)

Field	Description
	10 Reserved 11 Reserved
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PCTL	Parking Control Determines the slave port's parking control. The low-power park feature results in an overall power savings if the slave port is not saturated; however, this forces an extra latency clock when any master tries to access the slave port while not in use because it is not parked on any master. 00 When no master makes a request, the arbiter parks the slave port on the master port defined by the PARK bit field 01 When no master makes a request, the arbiter parks the slave port on the last master to be in control of the slave port 10 Low-power park. When no master makes a request, the slave port is not parked on a master and the arbiter drives all outputs to a constant safe state 11 Reserved
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 PARK	Park Determines which master port the current slave port parks on when no masters are actively making requests and the PCTL bits are cleared. **NOTE:** Select only master ports that are present on the chip. Otherwise, undefined behavior might occur. 000 Park on master port M0 001 Park on master port M1 010 Park on master port M2 011 Park on master port M3 100 Park on master port M4 101 Park on master port M5 110 Park on master port M6 111 Park on master port M7

16.3.3 Master General Purpose Control Register (AXBS_MGPCR*n*)

The MGPCR controls only whether the master's undefined length burst accesses are allowed to complete uninterrupted or whether they can be broken by requests from higher priority masters. The MGPCR can be accessed only in Supervisor mode with 32-bit accesses.

NOTE

If there are fewer than eight master ports, only the registers associated with those masters are present. Register addresses associated with master ports that are not present are reserved.

Please see this module's chip-specific information for which master ports are present on this module.

Address: 0h base + 800h offset + (256d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																													AULB		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AXBS_MGPCR*n* field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 AULB	Arbitrates On Undefined Length Bursts Determines whether, and when, the crossbar switch arbitrates away the slave port the master owns when the master is performing undefined length burst accesses. 000 No arbitration is allowed during an undefined length burst 001 Arbitration is allowed at any time during an undefined length burst 010 Arbitration is allowed after four beats of an undefined length burst 011 Arbitration is allowed after eight beats of an undefined length burst 100 Arbitration is allowed after 16 beats of an undefined length burst 101 Reserved 110 Reserved 111 Reserved

16.4 Functional Description

16.4.1 General operation

When a master accesses the crossbar switch, the access is immediately taken. If the targeted slave port of the access is available, then the access is immediately presented on the slave port. Single-clock or zero-wait-state accesses are possible through the crossbar. If the targeted slave port of the access is busy or parked on a different master port, the requesting master simply sees wait states inserted until the targeted slave port can service the master's request. The latency in servicing the request depends on each master's priority level and the responding slave's access time.

Because the crossbar switch appears to be just another slave to the master device, the master device has no knowledge of whether it actually owns the slave port it is targeting. While the master does not have control of the slave port it is targeting, it simply waits.

After the master has control of the slave port it is targeting, the master remains in control of the slave port until it relinquishes the slave port by running an IDLE cycle or by targeting a different slave port for its next access.

The master can also lose control of the slave port if another higher-priority master makes a request to the slave port. However, if the master is running a fixed- or undefined-length burst transfer it retains control of the slave port until that transfer completes. Based on MGPCR[AULB], the master either retains control of the slave port when doing undefined-length, incrementing burst transfers or loses the bus to a higher-priority master.

The crossbar terminates all master IDLE transfers, as opposed to allowing the termination to come from one of the slave buses. Additionally, when no master is requesting access to a slave port, the crossbar drives IDLE transfers onto the slave bus, even though a default master may be granted access to the slave port.

When a slave bus is being idled by the crossbar, it can park the slave port on the master port indicated by CRS*n*[PARK] . This is done to save the initial clock of arbitration delay that otherwise would be seen if the same master had to arbitrate to gain control of the slave port. The slave port can also be put into Low Power Park mode to save power, by using CRS*n*[PCTL].

16.4.2 Register coherency

The operation of the crossbar is affected as soon as a register is written. The values of the registers do not track with slave-port-related master accesses, but instead track only with slave accesses.

The MGPCRx[AULB] bits are the exception to this rule. The update of these bits is only recognized when the master on that master port runs an IDLE cycle, even though the slave bus cycle to write them will have already terminated successfully. If the MGPCRx[AULB] bits are written between two burst accesses, the new AULB encodings do not take effect until an IDLE cycle is initiated by the master on that master port.

16.4.3 Arbitration

The crossbar switch supports two arbitration algorithms:

- Fixed priority
- Round-robin

The arbitration scheme is independently programmable for each slave port.

16.4.3.1 Arbitration during undefined length bursts

Arbitration points during an undefined length burst are defined by the current master's MGPCR[AULB] field setting. When a defined length is imposed on the burst via the AULB bits, the undefined length burst is treated as a single or series of single back-to-back fixed-length burst accesses.

The following figure illustrates an example:

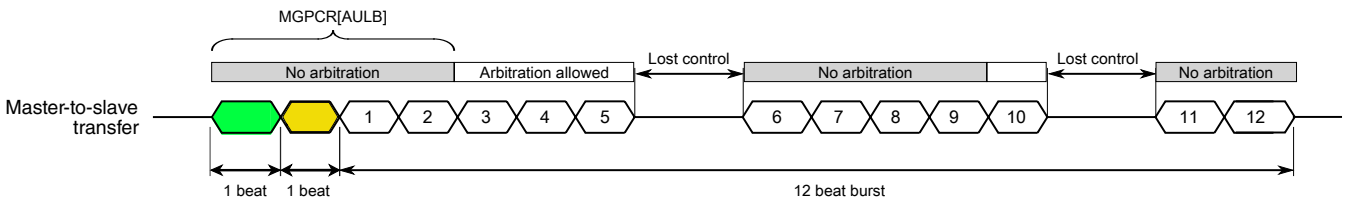


Figure 16-2. Undefined length burst example

In this example, a master runs an undefined length burst and the MGPCR[AULB] bits indicate arbitration occurs after the fourth beat of the burst. The master runs two sequential beats and then starts what will be a 12-beat undefined length burst access to a new address within the same slave port region as the previous access. The crossbar does not allow an arbitration point until the fourth overall access, or the second beat of the second burst. At that point, all remaining accesses are open for arbitration until the master loses control of the slave port.

Assume the master loses control of the slave port after the fifth beat of the second burst. After the master regains control of the slave port no arbitration point is available until after the master has run four more beats of its burst. After the fourth beat of the now continued burst, or the ninth beat of the second burst from the master's perspective, is taken, all beats of the burst are once again open for arbitration until the master loses control of the slave port.

Assume the master again loses control of the slave port on the fifth beat of the third now continued burst, or the 10th beat of the second burst from the master's perspective. After the master regains control of the slave port, it is allowed to complete its final two beats of its burst without facing arbitration.

Note

Fixed-length burst accesses are not affected by the AULB bits. All fixed-length burst accesses lock out arbitration until the last beat of the fixed-length burst.

16.4.3.2 Fixed-priority operation

When operating in fixed-priority mode, each master is assigned a unique priority level in the priority registers (PRS*n*). If two masters request access to the same slave port, the master with the highest priority in the selected priority register gains control over the slave port.

NOTE

In this arbitration mode, a higher-priority master can monopolize a slave port, preventing accesses from any lower-priority master to the port.

When a master makes a request to a slave port, the slave port checks whether the new requesting master's priority level is higher than that of the master that currently has control over the slave port, unless the slave port is in a parked state. The slave port performs an arbitration check at every clock edge to ensure that the proper master, if any, has control of the slave port.

The following table describes possible scenarios based on the requesting master port:

Table 16-2. How the Crossbar Switch grants control of a slave port to a master

When	Then the Crossbar Switch grants control to the requesting master
Both of the following are true: • The current master is not running a transfer. • The new requesting master's priority level is higher than that of the current master.	At the next clock edge
Both of the following are true: • The current master is running a fixed length burst transfer or a locked transfer. • The requesting master's priority level is higher than that of the current master.	At the end of the burst transfer or locked transfer
Both of the following are true: • The current master is running an undefined length burst transfer. • The requesting master's priority level is higher than that of the current master.	At the next arbitration point for the undefined length burst transfer **NOTE:** Arbitration points for an undefined length burst are defined in the MGPCR for each master.
The requesting master's priority level is lower than the current master.	At the conclusion of one of the following cycles: • An IDLE cycle • A non-IDLE cycle to a location other than the current slave port

16.4.3.3 Round-robin priority operation

When operating in round-robin mode, each master is assigned a relative priority based on the master port number. This relative priority is compared to the master port number (ID) of the last master to perform a transfer on the slave bus. The highest priority requesting master becomes owner of the slave bus at the next transfer boundary, accounting for locked and fixed-length burst transfers. Priority is based on how far ahead the ID of the requesting master is to the ID of the last master.

After granted access to a slave port, a master may perform as many transfers as desired to that port until another master makes a request to the same slave port. The next master in line is granted access to the slave port at the next transfer boundary, or possibly on the next clock cycle if the current master has no pending access request.

As an example of arbitration in round-robin mode, assume the crossbar is implemented with master ports 0, 1, 4, and 5. If the last master of the slave port was master 1, and master 0, 4, and 5 make simultaneous requests, they are serviced in the order: 4 then 5 then 0.

Parking may continue to be used in a round-robin mode, but does not affect the round-robin pointer unless the parked master actually performs a transfer. Handoff occurs to the next master in line after one cycle of arbitration. If the slave port is put into low-power park mode, the round-robin pointer is reset to point at master port 0, giving it the highest priority.

16.4.3.4 Priority assignment

Each master port must be assigned a unique 3-bit priority level. If an attempt is made to program multiple master ports with the same priority level within the priority registers (PRS*n*), the crossbar switch responds with a bus error and the registers are not updated.

16.5 Initialization/application information

No initialization is required for the crossbar switch.

Hardware reset ensures that all the register bits used by the crossbar switch are properly initialized to a valid state. However, settings and priorities may be programmed to achieve maximum system performance.

NOTE

- During configuration of the crossbar switch, all other masters must be idle.
- To prevent reconfiguration of the crossbar switch, write 1 to CRS*n*[RO].

Chapter 17
Peripheral Bridge

17.1 Chip-specific PBridge information

17.1.1 PBRIDGE Configuration

The peripheral bridge converts the crossbar switch interface to an interface that can access a majority of peripherals on the device.

This device contains two identical peripheral bridges: PBRIDGE_A (or AIPS_0) and PBRIDGE_B (or AIPS_1).

For SMPU Logical Bus Master Number, see table Table 20-1.

17.1.2 Register reset values

The following table shows the device-specific register reset values.

Table 17-1. AIPS0 register reset values

Register	Reset value
Master Privilege Register A (AIPS0_MPRA)	0x7775_0000
Master Privilege Register B (AIPS0_MPRB)	0x7770_0000
Peripheral Access Control Register A (AIPS0_PACRA)	0x5400_4000
Peripheral Access Control Register B (AIPS0_PACRB)	0x4440_4004
Peripheral Access Control Register C (AIPS0_PACRC)	0x4000_4400
Peripheral Access Control Register D (AIPS0_PACRD)	0x0044_0000
Peripheral Access Control Register E (AIPS0_PACRE)	0x0000_0000
Peripheral Access Control Register F (AIPS0_PACRF)	0x4000_0000
Peripheral Access Control Register G (AIPS0_PACRG)	0x0000_0000
Peripheral Access Control Register H (AIPS0_PACRH)	0x0000_0000

Table continues on the next page...

Table 17-1. AIPS0 register reset values (continued)

Register	Reset value
Off-platform Peripheral Access Control Register A (AIPS0_OPACRA)	0x4444_4045
Off-platform Peripheral Access Control Register B (AIPS0_OPACRB)	0x0004_4004
Off-platform Peripheral Access Control Register C (AIPS0_OPACRC)	0x0404_0404
Off-platform Peripheral Access Control Register D (AIPS0_OPACRD)	0x4440_0040
Off-platform Peripheral Access Control Register E (AIPS0_OPACRE)	0x0040_4040
Off-platform Peripheral Access Control Register F (AIPS0_OPACRF)	0x0004_0040
Off-platform Peripheral Access Control Register G (AIPS0_OPACRG)	0x4044_4000
Off-platform Peripheral Access Control Register H (AIPS0_OPACRH)	0x0000_0000
Off-platform Peripheral Access Control Register I (AIPS0_OPACRI)	0x0000_0000
Off-platform Peripheral Access Control Register J (AIPS0_OPACRJ)	0x0000_4444
Off-platform Peripheral Access Control Register K (AIPS0_OPACRK)	0x0000_0444
Off-platform Peripheral Access Control Register L (AIPS0_OPACRL)	0x4444_4000
Off-platform Peripheral Access Control Register M (AIPS0_OPACRM)	0x0044_4400
Off-platform Peripheral Access Control Register N (AIPS0_OPACRN)	0x0004_0000
Off-platform Peripheral Access Control Register O (AIPS0_OPACRO)	0x0000_0000
Off-platform Peripheral Access Control Register P (AIPS0_OPACRP)	0x0000_0004
Off-platform Peripheral Access Control Register Q (AIPS0_OPACRQ)	0x0000_0000
Off-platform Peripheral Access Control Register R (AIPS0_OPACRR)	0x0000_0000
Off-platform Peripheral Access Control Register S (AIPS0_OPACRS)	0x0000_0000
Off-platform Peripheral Access Control Register T (AIPS0_OPACRT)	0x0000_0000
Off-platform Peripheral Access Control Register U (AIPS0_OPACRU)	0x0000_0000
Off-platform Peripheral Access Control Register V (AIPS0_OPACRV)	0x0000_0000
Off-platform Peripheral Access Control Register W (AIPS0_OPACRW)	0x0000_0000

Table continues on the next page...

Table 17-1. AIPS0 register reset values (continued)

Register	Reset value
Off-platform Peripheral Access Control Register X (AIPS0_OPACRX)	0x0000_0000
Off-platform Peripheral Access Control Register Y (AIPS0_OPACRY)	0x0000_0044
Off-platform Peripheral Access Control Register Z (AIPS0_OPACRZ)	0x4000_0040
Off-platform Peripheral Access Control Register AA (AIPS0_OPACRAA)	0x0440_0000
Off-platform Peripheral Access Control Register AB (AIPS0_OPACRAB)	0x0000_0000
Off-platform Peripheral Access Control Register AC (AIPS0_OPACRAC)	0x0000_0000
Off-platform Peripheral Access Control Register AD (AIPS0_OPACRAD)	0x0000_0000
Off-platform Peripheral Access Control Register AE (AIPS0_OPACRAE)	0x0444_0044
Off-platform Peripheral Access Control Register AF (AIPS0_OPACRAF)	0x0004_0000

Table 17-2. AIPS1 register reset values

Register	Reset value
Master Privilege Register A (AIPS1_MPRA)	0x7775_0000
Master Privilege Register B (AIPS1_MPRB)	0x7770_0000
Peripheral Access Control Register A (AIPS1_PACRA)	0x5000_0000
Peripheral Access Control Register B (AIPS1_PACRB)	0x0000_0000
Peripheral Access Control Register C (AIPS1_PACRC)	0x0000_0000
Peripheral Access Control Register D (AIPS1_PACRD)	0x0000_0000
Peripheral Access Control Register E (AIPS1_OPACRE)	0x0000_0000
Peripheral Access Control Register F (AIPS1_OPACRF)	0x0000_0000
Peripheral Access Control Register G (AIPS1_OPACRG)	0x0000_0000
Peripheral Access Control Register H (AIPS1_OPACRH)	0x0000_0000
Off-platform Peripheral Access Control Register A (AIPS1_OPACRA)	0x0000_0000
Off-platform Peripheral Access Control Register B (AIPS1_OPACRB)	0x0000_0000
Off-platform Peripheral Access Control Register C (AIPS1_OPACRC)	0x0000_0004
Off-platform Peripheral Access Control Register D (AIPS1_OPACRD)	0x4000_0000
Off-platform Peripheral Access Control Register E (AIPS1_OPACRE)	0x0000_0000
Off-platform Peripheral Access Control Register F (AIPS1_OPACRF)	0x0400_0000

Table continues on the next page...

Table 17-2. AIPS1 register reset values (continued)

Register	Reset value
Off-platform Peripheral Access Control Register G (AIPS1_OPACRG)	0x0000_0000
Off-platform Peripheral Access Control Register H (AIPS1_OPACRH)	0x0000_0000
Off-platform Peripheral Access Control Register I (AIPS1_OPACRI)	0x0000_0000
Off-platform Peripheral Access Control Register J (AIPS1_OPACRJ)	0x0000_4444
Off-platform Peripheral Access Control Register K (AIPS1_OPACRK)	0x0000_0444
Off-platform Peripheral Access Control Register L (AIPS1_OPACRL)	0x4444_4000
Off-platform Peripheral Access Control Register M (AIPS1_OPACRM)	0x0044_4400
Off-platform Peripheral Access Control Register N (AIPS1_OPACRN)	0x0000_0000
Off-platform Peripheral Access Control Register O (AIPS1_OPACRO)	0x0000_0000
Off-platform Peripheral Access Control Register P (AIPS1_OPACRP)	0x0000_0004
Off-platform Peripheral Access Control Register Q (AIPS1_OPACRQ)	0x0000_0000
Off-platform Peripheral Access Control Register R (AIPS1_OPACRR)	0x0000_0000
Off-platform Peripheral Access Control Register S (AIPS1_OPACRS)	0x0000_0000
Off-platform Peripheral Access Control Register T (AIPS1_OPACRT)	0x0000_0000
Off-platform Peripheral Access Control Register U (AIPS1_OPACRU)	0x0000_0000
Off-platform Peripheral Access Control Register V (AIPS1_OPACRV)	0x0000_0000
Off-platform Peripheral Access Control Register W (AIPS1_OPACRW)	0x0000_0000
Off-platform Peripheral Access Control Register X (AIPS1_OPACRX)	0x0000_0000
Off-platform Peripheral Access Control Register Y (AIPS1_OPACRY)	0x0000_0000
Off-platform Peripheral Access Control Register Z (AIPS1_OPACRZ)	0x0000_0000
Off-platform Peripheral Access Control Register AA (AIPS1_OPACRAA)	0x0040_0000
Off-platform Peripheral Access Control Register AB (AIPS1_OPACRAB)	0x0000_0000
Off-platform Peripheral Access Control Register AC (AIPS1_OPACRAC)	0x0000_0000

Table continues on the next page...

Table 17-2. AIPS1 register reset values (continued)

Register	Reset value
Off-platform Peripheral Access Control Register AD (AIPS1_OPACRAD)	0x0000_0000
Off-platform Peripheral Access Control Register AE (AIPS1_OPACRAE)	0x0400_4400
Off-platform Peripheral Access Control Register AF (AIPS1_OPACRAF)	0x4400_0000

17.2 Introduction

The peripheral bridge converts the crossbar switch interface to an interface that can access most of the slave peripherals on this chip.

The peripheral bridge occupies 64 MB of the address space, which is divided into peripheral slots of 16 KB. (It might be possible that all the peripheral slots are not used. See the memory map chapter for details on slot assignments.) The bridge includes separate clock enable inputs for each of the slots to accommodate slower peripherals.

17.2.1 Features

Key features of the peripheral bridge are:

- Supports peripheral slots with 8-, 16-, and 32-bit datapath width
- Supports a pair of 32-bit transactions for selected 64-bit memory accesses
- Programming model provides memory protection functionality

17.2.2 General operation

The slave devices connected to the peripheral bridge are modules which contain a programming model of control and status registers. The system masters read and write these registers through the peripheral bridge.

The register maps of the peripherals are located on 16 KB boundaries. Each peripheral is allocated one or more 16-KB block(s) of the memory map.

17.3 Memory map/register definition

The 32-bit peripheral bridge registers can be accessed only in supervisor mode by trusted bus masters. Additionally, these registers must be read from or written to only by a 32-bit aligned access. The peripheral bridge registers are mapped into the Peripheral Access Control Register A PACRA[PACR0] address space.

AIPS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Master Privilege Register A (AIPS_MPRA)	32	R/W	See section	17.3.1/402
4	Master Privilege Register B (AIPS_MPRB)	32	R/W	See section	17.3.2/405
100	Peripheral Access Control Register (AIPS_PACRA)	32	R/W	4444_4444h	17.3.3/407
104	Peripheral Access Control Register (AIPS_PACRB)	32	R/W	4444_4444h	17.3.3/407
108	Peripheral Access Control Register (AIPS_PACRC)	32	R/W	4444_4444h	17.3.3/407
10C	Peripheral Access Control Register (AIPS_PACRD)	32	R/W	4444_4444h	17.3.3/407
110	Peripheral Access Control Register (AIPS_PACRE)	32	R/W	4444_4444h	17.3.3/407
114	Peripheral Access Control Register (AIPS_PACRF)	32	R/W	4444_4444h	17.3.3/407
118	Peripheral Access Control Register (AIPS_PACRG)	32	R/W	4444_4444h	17.3.3/407
11C	Peripheral Access Control Register (AIPS_PACRH)	32	R/W	4444_4444h	17.3.3/407
140	Off-Platform Peripheral Access Control Register (AIPS_OPACRA)	32	R/W	See section	17.3.4/412
144	Off-Platform Peripheral Access Control Register (AIPS_OPACRB)	32	R/W	See section	17.3.4/412
148	Off-Platform Peripheral Access Control Register (AIPS_OPACRC)	32	R/W	See section	17.3.4/412
14C	Off-Platform Peripheral Access Control Register (AIPS_OPACRD)	32	R/W	See section	17.3.4/412
150	Off-Platform Peripheral Access Control Register (AIPS_OPACRE)	32	R/W	See section	17.3.4/412
154	Off-Platform Peripheral Access Control Register (AIPS_OPACRF)	32	R/W	See section	17.3.4/412
158	Off-Platform Peripheral Access Control Register (AIPS_OPACRG)	32	R/W	See section	17.3.4/412
15C	Off-Platform Peripheral Access Control Register (AIPS_OPACRH)	32	R/W	See section	17.3.4/412
160	Off-Platform Peripheral Access Control Register (AIPS_OPACRI)	32	R/W	See section	17.3.4/412
164	Off-Platform Peripheral Access Control Register (AIPS_OPACRJ)	32	R/W	See section	17.3.4/412
168	Off-Platform Peripheral Access Control Register (AIPS_OPACRK)	32	R/W	See section	17.3.4/412

Table continues on the next page...

AIPS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
16C	Off-Platform Peripheral Access Control Register (AIPS_OPACRL)	32	R/W	See section	17.3.4/412
170	Off-Platform Peripheral Access Control Register (AIPS_OPACRM)	32	R/W	See section	17.3.4/412
174	Off-Platform Peripheral Access Control Register (AIPS_OPACRN)	32	R/W	See section	17.3.4/412
178	Off-Platform Peripheral Access Control Register (AIPS_OPACRO)	32	R/W	See section	17.3.4/412
17C	Off-Platform Peripheral Access Control Register (AIPS_OPACRP)	32	R/W	See section	17.3.4/412
180	Off-Platform Peripheral Access Control Register (AIPS_OPACRQ)	32	R/W	See section	17.3.4/412
184	Off-Platform Peripheral Access Control Register (AIPS_OPACRR)	32	R/W	See section	17.3.4/412
188	Off-Platform Peripheral Access Control Register (AIPS_OPACRS)	32	R/W	See section	17.3.4/412
18C	Off-Platform Peripheral Access Control Register (AIPS_OPACRT)	32	R/W	See section	17.3.4/412
190	Off-Platform Peripheral Access Control Register (AIPS_OPACRU)	32	R/W	See section	17.3.4/412
194	Off-Platform Peripheral Access Control Register (AIPS_OPACRV)	32	R/W	See section	17.3.4/412
198	Off-Platform Peripheral Access Control Register (AIPS_OPACRW)	32	R/W	See section	17.3.4/412
19C	Off-Platform Peripheral Access Control Register (AIPS_OPACRX)	32	R/W	See section	17.3.4/412
1A0	Off-Platform Peripheral Access Control Register (AIPS_OPACRY)	32	R/W	See section	17.3.4/412
1A4	Off-Platform Peripheral Access Control Register (AIPS_OPACRZ)	32	R/W	See section	17.3.4/412
1A8	Off-Platform Peripheral Access Control Register (AIPS_OPACRAA)	32	R/W	See section	17.3.4/412
1AC	Off-Platform Peripheral Access Control Register (AIPS_OPACRAB)	32	R/W	See section	17.3.4/412
1B0	Off-Platform Peripheral Access Control Register (AIPS_OPACRAC)	32	R/W	See section	17.3.4/412
1B4	Off-Platform Peripheral Access Control Register (AIPS_OPACRAD)	32	R/W	See section	17.3.4/412
1B8	Off-Platform Peripheral Access Control Register (AIPS_OPACRAE)	32	R/W	See section	17.3.4/412
1BC	Off-Platform Peripheral Access Control Register (AIPS_OPACRAF)	32	R/W	See section	17.3.4/412

17.3.1 Master Privilege Register A (AIPS_MPRA)

This register specifies eight identical 4-bit fields defining the access-privilege level associated with a bus master in the device to various peripherals. The register provides one field per bus master. Each master is assigned a logical master number. See the device configuration information for details.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MTR0	MTW0	MPL0	0	MTR1	MTW1	MPL1	Reserved				0	MTR3	MTW3	MPL3
W																
Reset	0*	1*	1*	1*	0*	0*	0*	0*	0	0	0	0	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	MTR4	MTW4	MPL4	0	MTR5	MTW5	MPL5	0	MTR6	MTW6	MPL6	Reserved			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0	0	0	0

* Notes:
- Reset value is chip-dependent. See the AIPS chip-specific information.

AIPS_MPRA field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 MTR0	Master 0 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
2 MTW0	Master 0 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
3 MPL0	Master 0 Privilege Level Specifies how the privilege level of the master is determined.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
	0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 MTR1	Master 1 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
6 MTW1	Master 1 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
7 MPL1	Master 1 privilege level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
8–11 Reserved	This field is reserved.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 MTR3	Master 3 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
14 MTW3	Master 3 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
15 MPL3	Master 3 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 MTR4	Master 4 Trusted For Read Determines whether the master is trusted for read accesses.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
	0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
18 MTW4	Master 4 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
19 MPL4	Master 4 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 MTR5	Master 5 Trusted For Read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
22 MTW5	Master 5 Trusted For Writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
23 MPL5	Master 5 Privilege Level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 MTR6	Master 6 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
26 MTW6	Master 6 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
27 MPL6	Master 6 privilege level Specifies how the privilege level of the master is determined.

Table continues on the next page...

AIPS_MPRA field descriptions (continued)

Field	Description
	0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
28–31 Reserved	This field is reserved.

17.3.2 Master Privilege Register B (AIPS_MPRB)

This register specifies eight identical 4-bit fields defining the access-privilege level associated with a bus master in the device to various peripherals. The register provides one field per bus master. Each master is assigned a logical master number. See the device configuration information for details.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	MTR8	MTW8	MPL8	0	MTR9	MTW9	MPL9	0	Reserved			0	Reserved		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0	0	0	0*	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	Reserved			0	Reserved			0	Reserved			0	Reserved		
W																
Reset	0*	0	0	0	0*	0	0	0	0*	0	0	0	0*	0	0	0

* Notes:
- Reset value is chip-dependent. See the AIPS chip-specific information.

AIPS_MPRB field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 MTR8	Master 8 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.

Table continues on the next page...

AIPS_MPRB field descriptions (continued)

Field	Description
2 MTW8	Master 8 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
3 MPL8	Master 8 privilege level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 MTR9	Master 9 trusted for read Determines whether the master is trusted for read accesses. 0 This master is not trusted for read accesses. 1 This master is trusted for read accesses.
6 MTW9	Master 9 trusted for writes Determines whether the master is trusted for write accesses. 0 This master is not trusted for write accesses. 1 This master is trusted for write accesses.
7 MPL9	Master 9 privilege level Specifies how the privilege level of the master is determined. 0 Accesses from this master are forced to user-mode. 1 Accesses from this master are not forced to user-mode.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–11 Reserved	This field is reserved.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 Reserved	This field is reserved.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 Reserved	This field is reserved.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 Reserved	This field is reserved.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

AIPS_MPRB field descriptions (continued)

Field	Description
25–27 Reserved	This field is reserved.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 Reserved	This field is reserved.

17.3.3 Peripheral Access Control Register (AIPS_PACR*n*)

Each of the on-platform peripherals has a 4-bit PACR*n* field, which defines the access levels supported by the given module. Eight PACR fields are grouped together to form a 32-bit PACR*x* register.

The peripheral assignment for each PACR field is defined by the memory map slot that the peripheral is assigned to. See the device's memory map details for the assignments for a particular device.

If a peripheral is absent, the corresponding PACR field is not implemented. Reads or writes to the location of an unimplemented PACR register (because all eight of its peripherals are absent) should be avoided to prevent undesired behavior.

The following table shows the top-level structure of PACRs.

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x100	PACRA	PACR0	PACR1	PACR2	PACR3	PACR4	PACR5	PACR6	PACR7
0x104	PACRB	PACR8	PACR9	PACR10	PACR11	PACR12	PACR13	PACR14	PACR15
0x108	PACRC	PACR16	PACR17	PACR18	PACR19	PACR20	PACR21	PACR22	PACR23
0x10C	PACRD	PACR24	PACR25	PACR26	PACR27	PACR28	PACR29	PACR30	PACR31
0x110	PACRE	PACR32	PACR33	PACR34	PACR35	PACR36	PACR37	PACR38	PACR39
0x114	PACRF	PACR40	PACR41	PACR42	PACR43	PACR44	PACR45	PACR46	PACR47
0x118	PACRG	PACR48	PACR49	PACR50	PACR51	PACR52	PACR53	PACR54	PACR55
0x11C	PACRH	PACR56	PACR57	PACR58	PACR59	PACR60	PACR61	PACR62	PACR63

NOTE

The reset value of PACR0 is 0101b.

Address: 0h base + 100h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SP0	WP0	TP0	0	SP1	WP1	TP1	0	SP2	WP2	TP2	0	SP3	WP3	TP3
W																
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SP4	WP4	TP4	0	SP5	WP5	TP5	0	SP6	WP6	TP6	0	SP7	WP7	TP7
W																
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPS_PACR*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 SP0	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
2 WP0	Write Protect Determines whether the peripheral allows write access. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
3 TP0	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SP1	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
6 WP1	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
7 TP1	Trusted Protect

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SP2	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
10 WP2	Write Protect Determines whether the peripheral allows write access. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
11 TP2	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SP3	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
14 WP3	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
15 TP3	Trusted Protect

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 SP4	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
18 WP4	Write Protect Determines whether the peripheral allows write access. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
19 TP4	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SP5	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
22 WP5	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
23 TP5	Trusted Protect

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SP6	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
26 WP6	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
27 TP6	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 SP7	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
30 WP7	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
31 TP7	Trusted Protect

Table continues on the next page...

AIPS_PACR*n* field descriptions (continued)

Field	Description
	Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

17.3.4 Off-Platform Peripheral Access Control Register (AIPS_OPACR*n*)

Each of the on-platform peripherals has a 4-bit OPACR*n* field, which defines the access levels supported by the given module. Eight OPACR fields are grouped together to form a 32-bit OPACR*x* register.

The peripheral assignment for each OPACR field is defined by the memory map slot that the peripheral is assigned to. See the device's memory map details for the assignments for a particular device.

If a peripheral is absent, the corresponding OPACR field is not implemented. Reads or writes to the location of an unimplemented OPACR register (because all eight of its peripherals are absent) should be avoided to prevent undesired behavior.

The following table shows the top-level structure of OPACRs.

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x140	OPACRA	OPACR0	OPACR1	OPACR2	OPACR3	OPACR4	OPACR5	OPACR6	OPACR7
0x144	OPACRB	OPACR8	OPACR9	OPACR10	OPACR11	OPACR12	OPACR13	OPACR14	OPACR15
0x148	OPACRC	OPACR16	OPACR17	OPACR18	OPACR19	OPACR20	OPACR21	OPACR22	OPACR23
0x14C	OPACRD	OPACR24	OPACR25	OPACR26	OPACR27	OPACR28	OPACR29	OPACR30	OPACR31
0x150	OPACRE	OPACR32	OPACR33	OPACR34	OPACR35	OPACR36	OPACR37	OPACR38	OPACR39
0x154	OPACRF	OPACR40	OPACR41	OPACR42	OPACR43	OPACR44	OPACR45	OPACR46	OPACR47
0x158	OPACRG	OPACR48	OPACR49	OPACR50	OPACR51	OPACR52	OPACR53	OPACR54	OPACR55
0x15C	OPACRH	OPACR56	OPACR57	OPACR58	OPACR59	OPACR60	OPACR61	OPACR62	OPACR63
0x160	OPACRI	OPACR64	OPACR65	OPACR66	OPACR67	OPACR68	OPACR69	OPACR70	OPACR71
0x164	OPACRJ	OPACR72	OPACR73	OPACR74	OPACR75	OPACR76	OPACR77	OPACR78	OPACR79
0x168	OPACRK	OPACR80	OPACR81	OPACR82	OPACR83	OPACR84	OPACR85	OPACR86	OPACR87
0x16C	OPACRL	OPACR88	OPACR89	OPACR90	OPACR91	OPACR92	OPACR93	OPACR94	OPACR95
0x170	OPACRM	OPACR96	OPACR97	OPACR98	OPACR99	OPACR100	OPACR101	OPACR102	OPACR103
0x174	OPACRN	OPACR104	OPACR105	OPACR106	OPACR107	OPACR108	OPACR109	OPACR110	OPACR111

Table continues on the next page...

Offset	Register	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
0x178	OPACRO	OPACR112	OPACR113	OPACR114	OPACR115	OPACR116	OPACR117	OPACR118	OPACR119
0x17C	OPACRP	OPACR120	OPACR121	OPACR122	OPACR123	OPACR124	OPACR125	OPACR126	OPACR127
0x180	OPACRQ	OPACR128	OPACR129	OPACR130	OPACR131	OPACR132	OPACR133	OPACR134	OPACR135
0x184	OPACRR	OPACR136	OPACR137	OPACR138	OPACR139	OPACR140	OPACR141	OPACR142	OPACR143
0x188	OPACRS	OPACR144	OPACR145	OPACR146	OPACR147	OPACR148	OPACR149	OPACR150	OPACR151
0x18C	OPACRT	OPACR152	OPACR153	OPACR154	OPACR155	OPACR156	OPACR157	OPACR158	OPACR159
0x190	OPACRU	OPACR160	OPACR161	OPACR162	OPACR163	OPACR164	OPACR165	OPACR166	OPACR167
0x194	OPACRV	OPACR168	OPACR169	OPACR170	OPACR171	OPACR172	OPACR173	OPACR174	OPACR175
0x198	OPACRW	OPACR176	OPACR177	OPACR178	OPACR179	OPACR180	OPACR181	OPACR182	OPACR183
0x19C	OPACRX	OPACR184	OPACR185	OPACR186	OPACR187	OPACR188	OPACR189	OPACR190	OPACR191
0x1A0	OPACRY	OPACR192	OPACR193	OPACR194	OPACR195	OPACR196	OPACR197	OPACR198	OPACR199
0x1A4	OPACRZ	OPACR200	OPACR201	OPACR202	OPACR203	OPACR204	OPACR205	OPACR206	OPACR207
0x1A8	OPACRAA	OPACR208	OPACR209	OPACR210	OPACR211	OPACR212	OPACR213	OPACR214	OPACR215
0x1AC	OPACRAB	OPACR216	OPACR217	OPACR218	OPACR219	OPACR220	OPACR221	OPACR222	OPACR223
0x1B0	OPACRAC	OPACR224	OPACR225	OPACR226	OPACR227	OPACR228	OPACR229	OPACR230	OPACR231
0x1B4	OPACRAD	OPACR232	OPACR233	OPACR234	OPACR235	OPACR236	OPACR237	OPACR238	OPACR239
0x1B8	OPACRAE	OPACR240	OPACR241	OPACR242	OPACR243	OPACR244	OPACR245	OPACR246	OPACR247
0x1BC	OPACRAF	OPACR248	OPACR249	OPACR250	OPACR251	OPACR252	OPACR253	OPACR254	OPACR255

Address: 0h base + 140h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	SP0	WP0	TP0	0	SP1	WP1	TP1	0	SP2	WP2	TP2	0	SP3	WP3	TP3
W																
Reset	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SP4	WP4	TP4	0	SP5	WP5	TP5	0	SP6	WP6	TP6	0	SP7	WP7	TP7
W																
Reset	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*	0*	1*	0*	0*

* Notes:

- Reset value is chip-dependent. See the AIPS chip-specific information.

AIPS_OPACR*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 SP0	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
2 WP0	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
3 TP0	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SP1	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
6 WP1	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
7 TP1	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
	0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SP2	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute MPR *x* [MPL *n*], and the MPR *x* [MPL *n*] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
10 WP2	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
11 TP2	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 SP3	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
14 WP3	Write Protect Determines whether the peripheral allows write access. When this bit is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
15 TP3	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
	0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 SP4	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for access. When this bit is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control bit for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
18 WP4	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
19 TP4	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this bit is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SP5	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
22 WP5	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
23 TP5	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
	0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 SP6	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
26 WP6	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
27 TP6	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates. 0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 SP7	Supervisor Protect Determines whether the peripheral requires supervisor privilege level for accesses. When this field is set, the master privilege level must indicate the supervisor access attribute, and the MPR *x* [MPL *n*] control field for the master must be set. If not, access terminates with an error response and no peripheral access initiates. 0 This peripheral does not require supervisor privilege level for accesses. 1 This peripheral requires supervisor privilege level for accesses.
30 WP7	Write Protect Determines whether the peripheral allows write accesses. When this field is set and a write access is attempted, access terminates with an error response and no peripheral access initiates. 0 This peripheral allows write accesses. 1 This peripheral is write protected.
31 TP7	Trusted Protect Determines whether the peripheral allows accesses from an untrusted master. When this field is set and an access is attempted by an untrusted master, the access terminates with an error response and no peripheral access initiates.

Table continues on the next page...

AIPS_OPACR*n* field descriptions (continued)

Field	Description
	0 Accesses from an untrusted master are allowed. 1 Accesses from an untrusted master are not allowed.

17.4 Functional description

The peripheral bridge functions as a bus protocol translator between the crossbar switch and the slave peripheral bus.

Support is provided for generating a pair of 32-bit slave accesses when performing certain 64-bit peripheral accesses.

The peripheral bridge manages all transactions destined for the attached slave devices and generates select signals for modules on the peripheral bus by decoding accesses within the attached address space.

17.4.1 Access support

All combinations of access size and peripheral data port width are supported. An access that is larger than the target peripheral's data width will be decomposed to multiple, smaller accesses. Bus decomposition is terminated by a transfer error caused by an access to an empty register area.

Chapter 18
Decorated Storage Memory Controller (DSMC)

18.1 Introduction

This section details the hardware support for atomic read-modify-write memory operations. In the Power Architecture, these capabilities are called "*decorated storage*". It is supported by capabilities in the processor cores plus instantiations of a Decorated Storage Memory Controller (DSMC).

The Decorated Storage APU defines instructions for providing load and store operations to memory addresses that require *additional semantics* beyond just the reading and writing of data values to the addressed memory locations. Decorated storage operations are intended to be used for specific devices or memory targets that require these additional semantics. A "decoration" is the additional semantic information to be applied to the decorated storage operation by the DSMC.

Consider the basic mnemonics, syntax and formats for the decorated load instructions.

For loads, the syntax is `l[b,h,w]d{cb}x rT,rB,rA` where the data size specifier is defined as 8-bit (`b` = byte), 16-bit (`h` = halfword) or 32-bit (`w` = word), and the three register specifiers include `rT` as the destination target register, `rB` as the effective address and `rA` as the decoration. The optional `{cb}` specifier defines a version of the instruction which is treated as a cache bypass operation.

The syntax for store instruction is `st[b,h,w]d{cb}x rS,rB,rA` where the same data size specifiers are used, and the three register specifiers are `rS` as the source data register, `rB` as the effective address and `rA` as the decoration. The cache bypass attribute is again specified with the use of the optional `{cb}` in the instruction mnemonic.

For all decorated loads and stores, the memory effective address is simply defined by the rB register (hence the `"x"` mnemonic suffix, signaling an "indexed" addressing mode) and the decoration is specified by the rA register. The core transmits the contents of the rA register as the decoration value along with the access address (register rB) to the DSMC.

The decorated load and store instruction formats are shown in the following figure.

Table 18-1. Power Architecture decorated load and store instruction formats

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
lbdx	0	1	1	1	1	1	rT						rA				rB				1	0	0	0	0	0	0	0	0	1	1	/
lhdx	0	1	1	1	1	1	rT						rA				rB				1	0	0	0	1	0	0	0	0	1	1	/
lwdx	0	1	1	1	1	1	rT						rA				rB				1	0	0	1	0	0	0	0	0	1	1	/
lbdcbx	0	1	1	1	1	1	rT						rA				rB				1	0	0	0	0	0	0	0	0	1	0	/
lhdcbx	0	1	1	1	1	1	rT						rA				rB				1	0	0	0	1	0	0	0	0	1	0	/
lwdcbx	0	1	1	1	1	1	rT						rA				rB				1	0	0	1	0	0	0	0	0	1	0	/
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
stbdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	0	0	0	0	0	1	1	/
sthdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	1	0	0	0	0	1	1	/
stwdx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	1	0	0	0	0	0	1	1	/
stbdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	0	0	0	0	0	1	0	/
sthdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	0	1	0	0	0	0	1	0	/
stwdcbx	0	1	1	1	1	1	**r**S						**r**A				**r**B				1	0	1	1	0	0	0	0	0	1	0	/

The basic decorated load or store memory reference supplements the access address and attribute information with a 32-bit "decoration". The DSMC decoration defines the special memory operation to be performed and optionally includes bit specifiers and/or operand data for the command.

The default format for the 32-bit decoration field defines a 4-bit command in the most significant bits (decoration[0:3]).

The cache bypass decoration load (`l[b,h,w]dcbx`) and store (`st[b,h,w]dcbx`) instructions do not distinguish between cacheable and cache-inhibited storage attributes, and instead have semantics to effectively emulate a cache-inhibited operation to storage regardless of the actual storage attributes. This is desired in order to allow a small portion of an otherwise cacheable storage area to be treated as non-cacheable using instruction supplied semantics, regardless of storage attributes provided by a memory management, protection or other memory control scheme. This allows decoration or other accesses to be applied to a portion of memory that is normally treated as cacheable according to its storage attributes, as well as to ensure that any stale copies of storage locations present in the cache are not used.

The combination of the decorated load and store instruction support in the e200zX cores plus the decorated storage memory controller in the core platform adds a robust atomic read-modify-write capability to the Power Architecture. The architecture capability defined by these core and platform functions is targeted at manipulation of *n*-bit fields in peripheral registers (and is consistent with I/O hardware addressing in the Embedded C standard) as well as software synchronization data structures needed in multi-core systems (mutexes, semaphores, test-and-set and compare-and-swap structures).

For additional information on the decorated load and store instructions from the core's perspective, refer to the appropriate e200zX core reference manual.

18.2 Decorated stores: `st[b,h,w]d{cb}x rS,rA,rB`

The next sections present descriptions of the specific operations, based on the 4-bit command field defined in decoration[0:3]. These descriptions include the bit pattern definitions for the 32-bit decoration value and include pseudo-code detailing the sequence of operations. The decoration formats are defined using a `<command>.<size>` syntax where the operand size specifier is b (byte, 8-bit), h (halfword, 16-bit) or w (word, 32-bit). The operand size is specified directly in the decorated instruction executed in the core, and this attribute is driven to the system bus as part of the decorated data transfer. Additionally, the write data is taken from the right-justified 8, 16 or 32 bits of the rS register, that is:

```
 8-bit wdata = rS[24:31]
16-bit wdata = rS[16:31]
32-bit wdata = rS[ 0:31]
```

Likewise, read data is loaded into the right-justified 8, 16 or 32 bits of the rT register:

```
 8-bit rdata = rT[24:31]
16-bit rdata = rT[16:31]
32-bit rdata = rT[ 0:31]
```

For the byte and halfword decorated load instructions, the upper bits of the rT register are zero filled.

The starting bit (SRTBIT) position follows the Power Architecture convention where the MSB is bit 0 and the LSB is bit 31. The bit field width (BFW) value defines the width and BFW = 0 specifies the maximum width, that is, the container size.

The basic decorated store includes three data transfer fields sourced from the core: the access address (rB), the decoration (rA) and the write data (wdata) operand (rS). There are five operations defined and most of these transactions convert a single core AHB write bus cycle into an atomic read-modify-write, that is, an indivisible read followed by write sequence. Support for three basic boolean logic functions (AND, OR, XOR) is provided along with a compare-and-store operation plus a bit field insert operation. These operations do not support any type of bit field wrapping.

In the next sections detailing the decorated store operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA

- accessAddress = rB
- wdata = rS

18.2.1 Bit Field Insert (BFINS) into an 8, 16 or 32-bit memory container

Table 18-2. Decoration format: BFINS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
bfins.b	0	0	0	0	0	0	0	SRTBIT			0	0	0	BFW			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.h	0	0	0	0	0	0	SRTBIT				0	0	BFW				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
bfins.w	0	0	0	0	0	SRTBIT					0	BFW					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command inserts a bit field defined by SRTBIT and BFW into the memory "container" defined by the access size associated with the decorated store instruction using an atomic read-modify-write sequence.

```
tmp = mem[accessAddress, size] // memory read
if bfw == 0
         then bfw = container

if ((srtbit + bfw) <= container) // generate bit mask
    mask = ((1 << bfw) - 1) << (container - (srtbit + bfw))
else
    mask = ((1 << bfw) - 1) >> ((srtbit + bfw) - container)

tmp = tmp  & ~mask// modify
    | wdata &  mask

mem[accessAddress, size] = tmp // memory write
```

The write data operand (wdata) associated with the decorated store instruction contains the bit field to be inserted. *It must be properly aligned within a right-justified container in the source register (rS)*, that is, within the lower 8 bits for a byte operation, the lower 16 bits for a halfword or the entire 32 bits for a word operation.

To illustrate, consider the following example of the insertion of the 3-bit field "xyz" into an 8-bit memory container, initially set to "abcd_efgh". For all cases, the bit field width (BFW) is 3.

```
if SRTBIT = 0 and the decorated store rS register[24:31] = xyz-_----,
   then destination is "xyzd_efgh"
if SRTBIT = 1 and the decorated store rS register[24:31] = -xyz_----,
   then destination is "axyz_efgh"
if SRTBIT = 2 and the decorated store rS register[24:31] = --xy_z---,
   then destination is "abxy_zfgh"
if SRTBIT = 3 and the decorated store rS register[24:31] = ---x_yz--,
   then destination is "abcx_yzgh"
if SRTBIT = 4 and the decorated store rS register[24:31] = ---_xyz-,
```

```
   then destination is "abcd_xyzh"
if SRTBIT = 5 and the decorated store rS register[24:31] = ---_-xyz,
   then destination is "abcd_exyz"
if SRTBIT = 6 and the decorated store rS register[24:31] = ---_--xy,
   then destination is "abcd_efxy"
if SRTBIT = 7 and the decorated store rS register[24:31] = ---_---x,
   then destination is "abcd_efgx"
```

If the bit field insert operation specifies SRTBIT as 0 and BFW as 0 (indicating the width matches the container width), then the operation is logically equivalent to a simple register store operation.

18.2.1.1 Compare-and-Store (CAST)

Table 18-3. Decoration format: CAST

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24 25 26 27	28 29 30 31
cast.b	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	CD8	
cast.h	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	CD16									
cast.w	1	0	0	1	CD28																					

This command begins by performing a read of the referenced memory address and comparing that data with the operand included in the decoration word. If the read data is equal to the compare operand, the write data associated with the decorated store instruction is placed into the memory location, else the original read data is rewritten into the memory location. The write operand is selected from the appropriate data byte lanes in the same manner as any memory store operation; this implies the write data operand is right justified in the rS source register.

```
tmp = mem[accessAddress, size]  // memory read

if (size == 8)             // define compare_operand
    compare_operand = CD8
else if (size == 16)
         compare_operand = CD16
     else compare_operand = {0x0, CD28}// zero-filled data

if (tmp == compare_operand) // compare - "modify"
     mem[accessAddress, size] = wdata// memory write
else mem[accessAddress, size] = tmp
```

Note for the word size operation (cast.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit compare operand.

18.2.1.1.1 Logical AND (AND)

Table 18-4. Decoration format: AND

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24 25 26 27	28 29 30 31

Table continues on the next page...

Table 18-4. Decoration format: AND (continued)

and.{b,h,w}	1	0	1	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical AND operation using the write operand defined by the rS register; finally, the result of the AND operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp & wdata      // modify
mem[accessAddress, size] = tmp// memory write
```

18.2.1.1.2 Logical OR (OR)

Table 18-5. Decoration format: OR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
or.{b,h,w}	1	1	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical OR operation using the write operand defined by the rS register; finally, the result of the OR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp | wdata          // modify
mem[accessAddress, size] = tmp// memory write
```

18.2.1.1.3 Logical Exclusive-OR (XOR)

Table 18-6. Decoration format: XOR

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
xor.{b,h,w}	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs an atomic read-modify-write of the referenced memory location. First, the location is read; it is then modified by performing a logical XOR operation using the write operand defined by the rS register; finally, the result of the XOR operation is written back into the referenced memory location.

```
tmp = mem[accessAddress, size] // memory read
tmp = tmp ^ wdata          // modify
mem[accessAddress, size] = tmp// memory write
```

18.2.1.2 Decorated Loads: `l[b,h,w]d{cb}x rT,rA,rB`

The basic decorated load includes two data transfer fields sourced from the core: the access address (rB), the decoration (rA) plus the read data (rdata) operand returned from DSMC to the core and loaded into the destination register (rT). There are 3 operations defined and two of these transactions convert a single core AHB read bus cycle into an atomic read-modify-write. Support for a swap function, a load-and-set-1-bit functions are provided along with a simple load (aka an "extract") operation.

In the next sections detailing the decorated load operations, the following associations between the pseudocode variables and the core registers apply.

- decoration = rA
- accessAddress = rB
- rdata = rT

18.2.1.2.1 Simple Load (SLD)

Table 18-7. Decoration format: SLD

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
sld. {b,h,w}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command performs a simple memory load of the reference size from the access address. For this operation, the 12-bit field defined in decoration[4:15] must be zeroes else the transfer is not performed and error terminated.

```
rdata = mem[accessAddress, size] // memory read
```

Support for generic bit field extract operations is best handled using existing core and compiler capabilities involving standard load instructions followed by left and right shift operations to emulate these functions.

18.2.1.2.2 Registers-and-memory exchange (SWAP)

Table 18-8. Decoration format: SWAP

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
swap.b	0	1	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	WD8							
swap.h	0	1	0	1	-	-	-	-	-	-	-	-	-	-	-	-	**WD16**															
swap.w	0	1	0	1	WD28																											

This command loads the contents of the referenced memory location into the core's rT register and stores the source operand from the core's rA register (the decoration) into the memory location.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp// rT = rdata = tmp

if (size == 8)             // store_operand - "modify"
    store_operand = WD8
else if (size == 16)
           store_operand = WD16
    else store_operand = {0x0, WD28} // zero-filled write data

mem[accessAddress, size] = store_operand// memory write
```

Note for the word size operation (swap.w), the data value specified in the low-order 28 bits of the decoration is zero filled in the most significant bits to create the required 32 bit store operand.

18.2.1.2.3 Load-and-Set-1(Bit) (LAS1)

Table 18-9. Decoration format: LAS1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
las1.b	0	1	1	0	0	0	0	BIT			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.h	0	1	1	0	0	0	BIT				-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
las1.w	0	1	1	0	0	BIT					-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

This command first reads the referenced memory location; it then modifies the read operand by setting the single bit position defined in the decoration (BIT); the modified data is then written back to the referenced memory location and the original read data returned to the initiating processor core.

```
tmp   = mem[accessAddress, size] // memory read
rdata = tmp // rT = rdata = tmp
mask  = 1 << (container - bit - 1) // generate bit mask
tmp   = tmp | mask// modify
mem[accessAddress, size] = tmp// memory write
```

18.3 DSMC instantiations

The decorated storage memory controller which performs these operations is instantiated multiple times within the core platform and physically resides between the slave ports of the crossbar and the targeted memory controllers.

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

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

Chapter 19
Intelligent AHB Gasket (IAHBG)

19.1 Chip-specific IAHBG information

19.1.1 IAHBG configuration

Table 19-1 shows the operating modes implemented in the IAHBG on this chip.

Table 19-1. IAHBG implemented operation modes

Operating mode (Master:Slave)	Comments
1:1	Master and slave at same operating frequency; gasket performance optimizations on by default
2:1	Master at twice frequency of slave; gasket performance optimizations on by default
1:2	Slave at twice frequency of master; gasket performance optimizations on by default
4:1	Master at four times frequency of slave; gasket performance optimizations disabled at this time
1:4	Slave at four times frequency of master; gasket performance optimizations disabled at this time

19.1.2 IAHBG clock selection ratios

The IAHBG is used as a bus gasket to connect buses running at different bus clock speeds. Internal signals generated by the MC_CGL are connected to the IAHBG module that determine the clock ratio selection. Clock ratio setting for the IAHBG is controlled using DFS configurations. Refer Dynamic Frequency Scaling for details.

19.1.3 IAHBG features enabled by Platform Configuration Module (PCM)

The Platform Configuration Module (PCM) is used to enable/disable features of the IAHBG module. In general, the default setting in the PCM enables IAHBG features for the highest performance case.

19.2 Introduction

This section details the intelligent bridging gasket between the fast and slow clock domains. The intelligent operation of the gasket initiates pending reads early and optimizes bursts to recover some of the performance that is lost on a registered interface between different clock domains. Pending reads begin even if there is not an hready to the master. Non-bursted writes only begin when the slave is IDLE, pending writes wait until IDLE.

The IAHBG complies with the AHBV6lite protocol, including an extension to support masters that do address retraction during the error termination sequence or abort a burst read due to an error termination or another non-error termination like cache inhibit termination.

Table 19-2 lists the available operation modes.

Table 19-2. Operation modes available

Operating mode (Master:Slave)	Comments
0:0	Flowthru (no register wall)
1:1	Master and slave at same operating frequency; gasket performance optimizations on by default
2:1	Master at twice frequency of slave; gasket performance optimizations on by default
1:2	Slave at twice frequency of master; gasket performance optimizations on by default
4:1	Master at four times frequency of slave; gasket performance optimizations disabled at this time
1:4	Slave at four times frequency of master; gasket performance optimizations disabled at this time

19.3 Timing modes

19.3.1 1:1

In the 1:1 timing mode the master is running at the same clock frequency as the slave. The pending read can either be a single or burst transaction.

19.3.2 2:1

In the 2:1 timing mode the master is running at twice the clock frequency of the slave. The 4:1 mode is not shown because it is very similar to the 2:1 mode. The pending read can either be a single or burst transaction.

19.3.3 1:2

In the 1:2 timing mode, the slave is running at twice the clock frequency of the master. The 1:4 mode is not shown because it is very similar to the 1:2 mode. The pending read can either be a single or burst transaction.

Because the slave is operating at a faster clock frequency than the master, the transactions from the master must be stored into a queue of data and response attribute registers. At a certain point during the master burst transactions, the queue is either emptied into the master for a burst read or it is emptied into the slave for a burst write..

Chapter 20
System Memory Protection Unit (SMPU)

20.1 Chip-specific SMPU information

20.1.1 System Memory Protection Unit (SMPU) configuration

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

- SMPU_0 protects flash, peripherals and RAM.

The following figure explains how SMPU is configured on this chip.

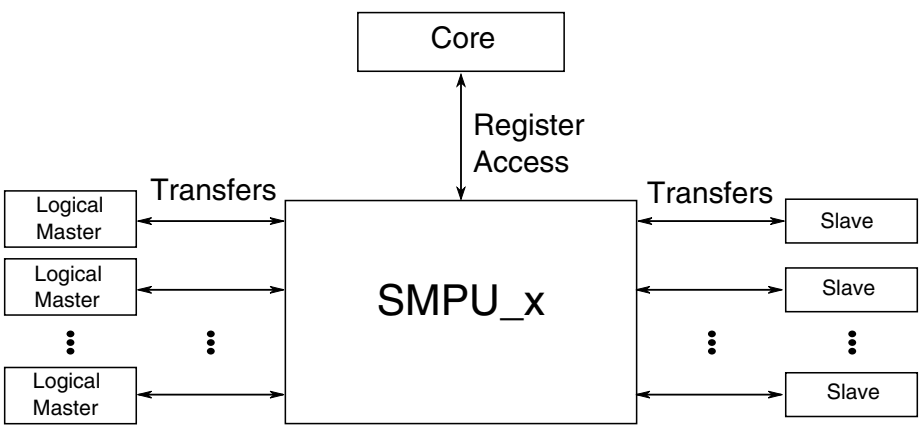


Figure 20-1. SMPU configuration

Table 20-1. SMPU Logical Bus Master Number

Master Name	Master ID
e200Z4	0
e200Z2	1
HSM	3
eDMA	4
Ethernet	5
FlexRAY	6

Table continues on the next page...

Table 20-1. SMPU Logical Bus Master Number (continued)

Master Name	Master ID
Debug e200Z4	8
Debug e200Z2	9

NOTE

Only Z4, Z2, and HSM core provide process identifier to SMPU.

NOTE

PID miss error does not occur for Debug e200Z4 and Debug e200Z2.

20.1.1.1 SMPU core exceptions

The core interrupt handling for a SMPU protected access is summarized in the following table:

SMPU Protected Access	Z4	Z2
Read	Data Storage Exception	Data Storage Exception
Write	See below	Data Storage Exception
Execute	Instruction Storage exception	Instruction Storage exception

The Z4 core interrupt handling for a SMPU protected write access is summarized in the following table:

MSR[ME]	HDBCR0[DSB]	Exception
0	0	No exception
0	1	Data Storage
1	0	Machine Check
1	1	Data Storage

The Z4 core has a store buffer and when enabled any core store to a SMPU protected region will be imprecise. For this scenario an imprecise data access will result in a machine check interrupt, when the machine check interrupt is enabled at Machine State Register (MSR). For the case when the data storage buffer is disabled the Z4 core will issue a data storage exception.

HDBCR0 is the hardware debug cache control register. This register would not normally be used by a normal user. It is intended to be used only by debugger vendors. Please refer to the e200z4 Core Reference manual for details.

20.2 Overview

The System Memory Protection Unit (SMPU) provides hardware access control for system bus memory references. The SMPU concurrently monitors system bus transactions and evaluates their appropriateness using preprogrammed region descriptors that define memory spaces and their access rights. Memory references that have sufficient access control rights are allowed to complete, while references that are not mapped to any region descriptor or have insufficient rights are terminated with an access error response.

20.2.1 Block diagram

A simplified block diagram of the SMPU module is shown in the following figure. The access signals coming from the crossbar switch slave ports are shown on the left. The access evaluation macros and region descriptors are in the middle, and the peripheral bus interface is on the right side.

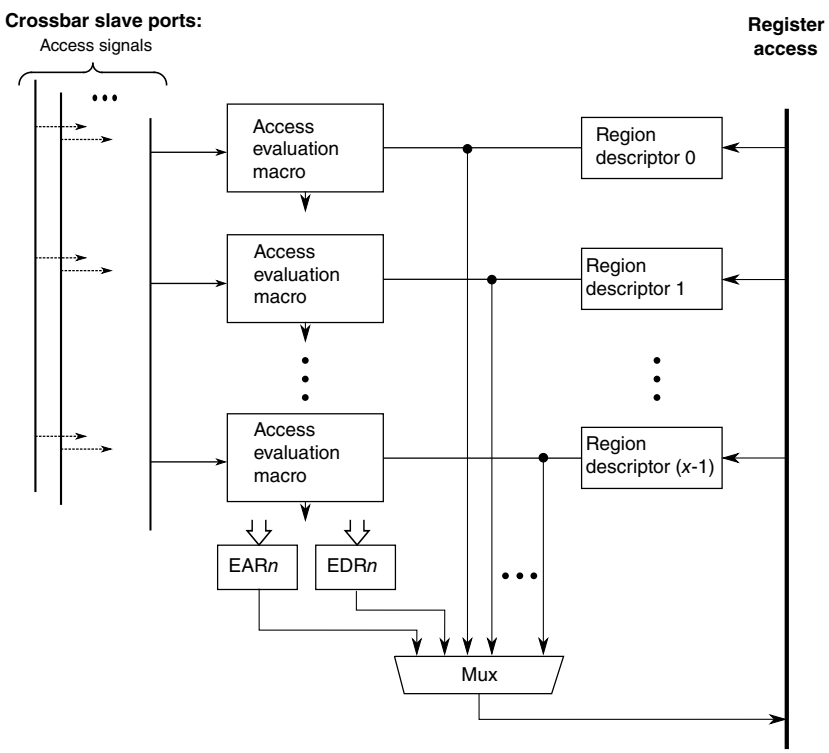


Figure 20-2. SMPU block diagram

The access evaluation macros determine hits to memory regions and detect violations to access protections. If a violation is detected, the macros update the appropriate master's error registers. For details of the access evaluation macro, see Access evaluation macro.

20.2.2 Features

The SMPU feature set includes:

- Support for a maximum of 16 program-visible 192-bit region descriptors, accessible as six 32-bit words each.
  - Each region descriptor defines an arbitrarily-sized space, aligned anywhere in memory.
    - Region sizes can vary from a minimum of 16 bytes to a maximum of (4 GB – 16 bytes).
  - Read/write access control permissions are defined in the region descriptor.
  - Global cache-inhibit attribute indicator.
  - Hardware-assisted maintenance of the descriptor valid bit minimizes coherency issues.
  - Priority given to granting permission over denying access for overlapping region descriptors.
- Detection of access errors if a memory reference does not hit in any memory region, or if the reference is illegal in all hit memory regions. If an access error occurs, the reference is terminated with an error response, and the SMPU inhibits the bus cycle being sent to the targeted slave device.
- Instruction storage (ISI) or data storage (DSI) interrupts generated by aborted core accesses.
- Error registers (per bus master ID) capture the last faulting address, attributes, and other information.
- Global SMPU enable/disable control bit.

20.3 Memory map/register definition

The programming model is partitioned into three groups:

- Control registers

- Error capture registers
- Data structure containing the region descriptors

The programming model can be referenced only in supervisor mode using 32-bit accesses. Attempted references with a different access size, to an undefined (reserved) address, with a non-supported access type (a write to a read-only register, or a read of a write-only register), or in user mode generate an error termination.

NOTE

See the chip configuration details for any chip-specific register information for this module. For example, a specific instance of the module may support only eight or fewer region descriptors.

SMPU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control/Error Status Register 0 (SMPU_CES0)	32	R/W	0000_8008h	20.3.1/442
4	Control/Error Status Register 1 (SMPU_CES1)	32	R/W	0000_8004h	20.3.2/443
100	Error Address Register, Bus Master (SMPU_ERR_ADR0)	32	R	0000_0000h	20.3.3/443
104	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL0)	32	R	0000_0000h	20.3.4/444
108	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL0)	32	R	0000_0080h	20.3.5/445
110	Error Address Register, Bus Master (SMPU_ERR_ADR1)	32	R	0000_0000h	20.3.3/443
114	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL1)	32	R	0000_0000h	20.3.4/444
118	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL1)	32	R	0000_0080h	20.3.5/445
120	Error Address Register, Bus Master (SMPU_ERR_ADR2)	32	R	0000_0000h	20.3.3/443
124	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL2)	32	R	0000_0000h	20.3.4/444
128	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL2)	32	R	0000_0080h	20.3.5/445
130	Error Address Register, Bus Master (SMPU_ERR_ADR3)	32	R	0000_0000h	20.3.3/443
134	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL3)	32	R	0000_0000h	20.3.4/444
138	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL3)	32	R	0000_0080h	20.3.5/445
140	Error Address Register, Bus Master (SMPU_ERR_ADR4)	32	R	0000_0000h	20.3.3/443
144	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL4)	32	R	0000_0000h	20.3.4/444
148	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL4)	32	R	0000_0080h	20.3.5/445
150	Error Address Register, Bus Master (SMPU_ERR_ADR5)	32	R	0000_0000h	20.3.3/443
154	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL5)	32	R	0000_0000h	20.3.4/444

Table continues on the next page...

SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
158	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL5)	32	R	0000_0080h	20.3.5/445
160	Error Address Register, Bus Master (SMPU_ERR_ADR6)	32	R	0000_0000h	20.3.3/443
164	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL6)	32	R	0000_0000h	20.3.4/444
168	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL6)	32	R	0000_0080h	20.3.5/445
170	Error Address Register, Bus Master (SMPU_ERR_ADR7)	32	R	0000_0000h	20.3.3/443
174	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL7)	32	R	0000_0000h	20.3.4/444
178	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL7)	32	R	0000_0080h	20.3.5/445
180	Error Address Register, Bus Master (SMPU_ERR_ADR8)	32	R	0000_0000h	20.3.3/443
184	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL8)	32	R	0000_0000h	20.3.4/444
188	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL8)	32	R	0000_0080h	20.3.5/445
190	Error Address Register, Bus Master (SMPU_ERR_ADR9)	32	R	0000_0000h	20.3.3/443
194	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL9)	32	R	0000_0000h	20.3.4/444
198	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL9)	32	R	0000_0080h	20.3.5/445
1A0	Error Address Register, Bus Master (SMPU_ERR_ADR10)	32	R	0000_0000h	20.3.3/443
1A4	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL10)	32	R	0000_0000h	20.3.4/444
1A8	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL10)	32	R	0000_0080h	20.3.5/445
1B0	Error Address Register, Bus Master (SMPU_ERR_ADR11)	32	R	0000_0000h	20.3.3/443
1B4	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL11)	32	R	0000_0000h	20.3.4/444
1B8	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL11)	32	R	0000_0080h	20.3.5/445
1C0	Error Address Register, Bus Master (SMPU_ERR_ADR12)	32	R	0000_0000h	20.3.3/443
1C4	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL12)	32	R	0000_0000h	20.3.4/444
1C8	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL12)	32	R	0000_0080h	20.3.5/445
1D0	Error Address Register, Bus Master (SMPU_ERR_ADR13)	32	R	0000_0000h	20.3.3/443
1D4	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL13)	32	R	0000_0000h	20.3.4/444
1D8	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL13)	32	R	0000_0080h	20.3.5/445
1E0	Error Address Register, Bus Master (SMPU_ERR_ADR14)	32	R	0000_0000h	20.3.3/443

Table continues on the next page...

SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1E4	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL14)	32	R	0000_0000h	20.3.4/444
1E8	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL14)	32	R	0000_0080h	20.3.5/445
1F0	Error Address Register, Bus Master (SMPU_ERR_ADR15)	32	R	0000_0000h	20.3.3/443
1F4	Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL15)	32	R	0000_0000h	20.3.4/444
1F8	Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL15)	32	R	0000_0080h	20.3.5/445
400	Region Descriptor n, Word 0 (SMPU_RGD0_WRD0)	32	R/W	0000_0000h	20.3.6/446
404	Region Descriptor n, Word 1 (SMPU_RGD0_WRD1)	32	R/W	0000_000Fh	20.3.7/447
408	Region Descriptor n, Word 2 (SMPU_RD_0W2)	32	R/W	0000_0000h	20.3.8/447
40C	Region Descriptor n, Word 3 (SMPU_RGD0_WRD3)	32	R/W	0000_0000h	20.3.9/449
410	Region Descriptor n, Word 4 (SMPU_RGD0_WRD4)	32	R/W	0000_0000h	20.3.10/450
414	Region Descriptor n, Word 5 (SMPU_RGD0_WRD5)	32	R/W	0000_0000h	20.3.11/451
440	Region Descriptor n, Word 0 (SMPU_RGD1_WRD0)	32	R/W	0000_0000h	20.3.6/446
444	Region Descriptor n, Word 1 (SMPU_RGD1_WRD1)	32	R/W	0000_000Fh	20.3.7/447
448	Region Descriptor n, Word 2 (SMPU_RD_1W2)	32	R/W	0000_0000h	20.3.8/447
44C	Region Descriptor n, Word 3 (SMPU_RGD1_WRD3)	32	R/W	0000_0000h	20.3.9/449
450	Region Descriptor n, Word 4 (SMPU_RGD1_WRD4)	32	R/W	0000_0000h	20.3.10/450
454	Region Descriptor n, Word 5 (SMPU_RGD1_WRD5)	32	R/W	0000_0000h	20.3.11/451
480	Region Descriptor n, Word 0 (SMPU_RGD2_WRD0)	32	R/W	0000_0000h	20.3.6/446
484	Region Descriptor n, Word 1 (SMPU_RGD2_WRD1)	32	R/W	0000_000Fh	20.3.7/447
488	Region Descriptor n, Word 2 (SMPU_RD_2W2)	32	R/W	0000_0000h	20.3.8/447
48C	Region Descriptor n, Word 3 (SMPU_RGD2_WRD3)	32	R/W	0000_0000h	20.3.9/449
490	Region Descriptor n, Word 4 (SMPU_RGD2_WRD4)	32	R/W	0000_0000h	20.3.10/450
494	Region Descriptor n, Word 5 (SMPU_RGD2_WRD5)	32	R/W	0000_0000h	20.3.11/451
4C0	Region Descriptor n, Word 0 (SMPU_RGD3_WRD0)	32	R/W	0000_0000h	20.3.6/446
4C4	Region Descriptor n, Word 1 (SMPU_RGD3_WRD1)	32	R/W	0000_000Fh	20.3.7/447
4C8	Region Descriptor n, Word 2 (SMPU_RD_3W2)	32	R/W	0000_0000h	20.3.8/447
4CC	Region Descriptor n, Word 3 (SMPU_RGD3_WRD3)	32	R/W	0000_0000h	20.3.9/449
4D0	Region Descriptor n, Word 4 (SMPU_RGD3_WRD4)	32	R/W	0000_0000h	20.3.10/450
4D4	Region Descriptor n, Word 5 (SMPU_RGD3_WRD5)	32	R/W	0000_0000h	20.3.11/451
500	Region Descriptor n, Word 0 (SMPU_RGD4_WRD0)	32	R/W	0000_0000h	20.3.6/446
504	Region Descriptor n, Word 1 (SMPU_RGD4_WRD1)	32	R/W	0000_000Fh	20.3.7/447
508	Region Descriptor n, Word 2 (SMPU_RD_4W2)	32	R/W	0000_0000h	20.3.8/447
50C	Region Descriptor n, Word 3 (SMPU_RGD4_WRD3)	32	R/W	0000_0000h	20.3.9/449
510	Region Descriptor n, Word 4 (SMPU_RGD4_WRD4)	32	R/W	0000_0000h	20.3.10/450
514	Region Descriptor n, Word 5 (SMPU_RGD4_WRD5)	32	R/W	0000_0000h	20.3.11/451
540	Region Descriptor n, Word 0 (SMPU_RGD5_WRD0)	32	R/W	0000_0000h	20.3.6/446

Table continues on the next page...

SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
544	Region Descriptor n, Word 1 (SMPU_RGD5_WRD1)	32	R/W	0000_000Fh	20.3.7/447
548	Region Descriptor n, Word 2 (SMPU_RD_5W2)	32	R/W	0000_0000h	20.3.8/447
54C	Region Descriptor n, Word 3 (SMPU_RGD5_WRD3)	32	R/W	0000_0000h	20.3.9/449
550	Region Descriptor n, Word 4 (SMPU_RGD5_WRD4)	32	R/W	0000_0000h	20.3.10/450
554	Region Descriptor n, Word 5 (SMPU_RGD5_WRD5)	32	R/W	0000_0000h	20.3.11/451
580	Region Descriptor n, Word 0 (SMPU_RGD6_WRD0)	32	R/W	0000_0000h	20.3.6/446
584	Region Descriptor n, Word 1 (SMPU_RGD6_WRD1)	32	R/W	0000_000Fh	20.3.7/447
588	Region Descriptor n, Word 2 (SMPU_RD_6W2)	32	R/W	0000_0000h	20.3.8/447
58C	Region Descriptor n, Word 3 (SMPU_RGD6_WRD3)	32	R/W	0000_0000h	20.3.9/449
590	Region Descriptor n, Word 4 (SMPU_RGD6_WRD4)	32	R/W	0000_0000h	20.3.10/450
594	Region Descriptor n, Word 5 (SMPU_RGD6_WRD5)	32	R/W	0000_0000h	20.3.11/451
5C0	Region Descriptor n, Word 0 (SMPU_RGD7_WRD0)	32	R/W	0000_0000h	20.3.6/446
5C4	Region Descriptor n, Word 1 (SMPU_RGD7_WRD1)	32	R/W	0000_000Fh	20.3.7/447
5C8	Region Descriptor n, Word 2 (SMPU_RD_7W2)	32	R/W	0000_0000h	20.3.8/447
5CC	Region Descriptor n, Word 3 (SMPU_RGD7_WRD3)	32	R/W	0000_0000h	20.3.9/449
5D0	Region Descriptor n, Word 4 (SMPU_RGD7_WRD4)	32	R/W	0000_0000h	20.3.10/450
5D4	Region Descriptor n, Word 5 (SMPU_RGD7_WRD5)	32	R/W	0000_0000h	20.3.11/451
600	Region Descriptor n, Word 0 (SMPU_RGD8_WRD0)	32	R/W	0000_0000h	20.3.6/446
604	Region Descriptor n, Word 1 (SMPU_RGD8_WRD1)	32	R/W	0000_000Fh	20.3.7/447
608	Region Descriptor n, Word 2 (SMPU_RD_8W2)	32	R/W	0000_0000h	20.3.8/447
60C	Region Descriptor n, Word 3 (SMPU_RGD8_WRD3)	32	R/W	0000_0000h	20.3.9/449
610	Region Descriptor n, Word 4 (SMPU_RGD8_WRD4)	32	R/W	0000_0000h	20.3.10/450
614	Region Descriptor n, Word 5 (SMPU_RGD8_WRD5)	32	R/W	0000_0000h	20.3.11/451
640	Region Descriptor n, Word 0 (SMPU_RGD9_WRD0)	32	R/W	0000_0000h	20.3.6/446
644	Region Descriptor n, Word 1 (SMPU_RGD9_WRD1)	32	R/W	0000_000Fh	20.3.7/447
648	Region Descriptor n, Word 2 (SMPU_RD_9W2)	32	R/W	0000_0000h	20.3.8/447
64C	Region Descriptor n, Word 3 (SMPU_RGD9_WRD3)	32	R/W	0000_0000h	20.3.9/449
650	Region Descriptor n, Word 4 (SMPU_RGD9_WRD4)	32	R/W	0000_0000h	20.3.10/450
654	Region Descriptor n, Word 5 (SMPU_RGD9_WRD5)	32	R/W	0000_0000h	20.3.11/451
680	Region Descriptor n, Word 0 (SMPU_RGD10_WRD0)	32	R/W	0000_0000h	20.3.6/446
684	Region Descriptor n, Word 1 (SMPU_RGD10_WRD1)	32	R/W	0000_000Fh	20.3.7/447
688	Region Descriptor n, Word 2 (SMPU_RD_10W2)	32	R/W	0000_0000h	20.3.8/447
68C	Region Descriptor n, Word 3 (SMPU_RGD10_WRD3)	32	R/W	0000_0000h	20.3.9/449
690	Region Descriptor n, Word 4 (SMPU_RGD10_WRD4)	32	R/W	0000_0000h	20.3.10/450
694	Region Descriptor n, Word 5 (SMPU_RGD10_WRD5)	32	R/W	0000_0000h	20.3.11/451
6C0	Region Descriptor n, Word 0 (SMPU_RGD11_WRD0)	32	R/W	0000_0000h	20.3.6/446
6C4	Region Descriptor n, Word 1 (SMPU_RGD11_WRD1)	32	R/W	0000_000Fh	20.3.7/447
6C8	Region Descriptor n, Word 2 (SMPU_RD_11W2)	32	R/W	0000_0000h	20.3.8/447
6CC	Region Descriptor n, Word 3 (SMPU_RGD11_WRD3)	32	R/W	0000_0000h	20.3.9/449

Table continues on the next page...

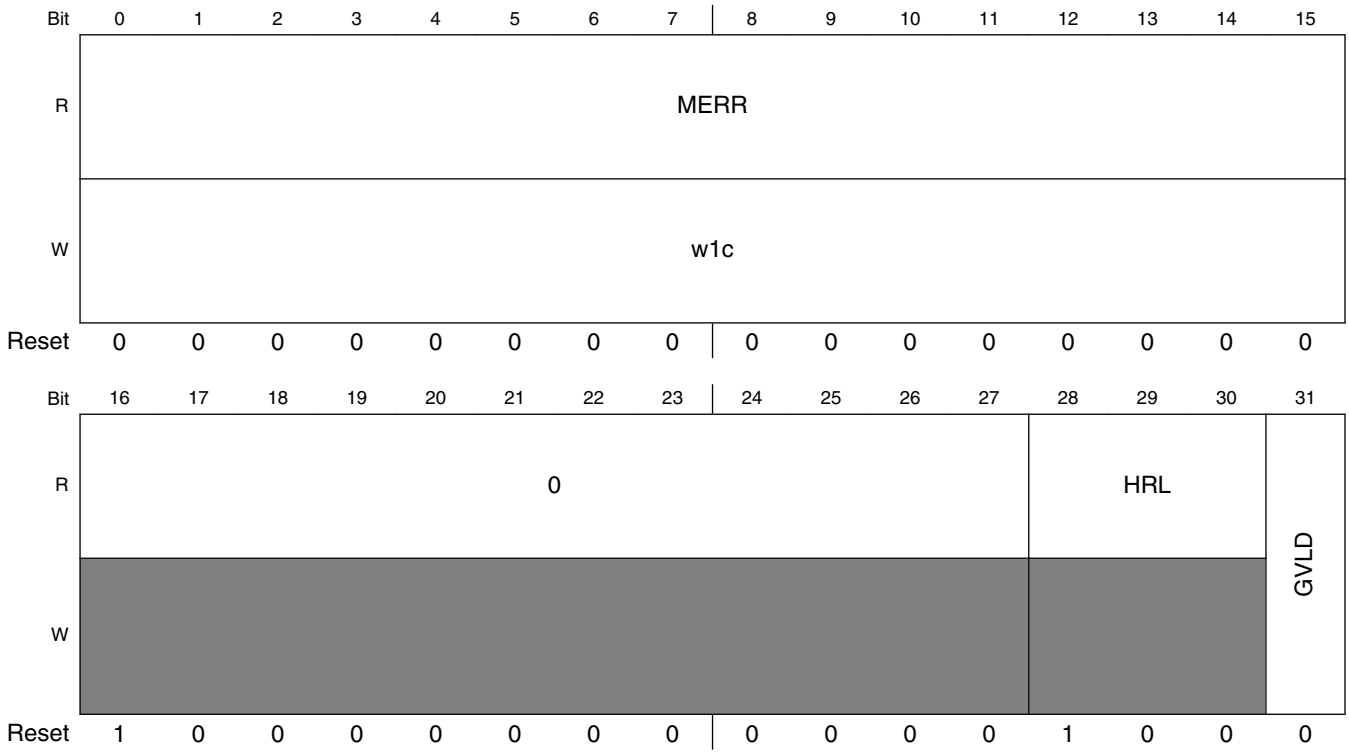
SMPU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6D0	Region Descriptor n, Word 4 (SMPU_RGD11_WRD4)	32	R/W	0000_0000h	20.3.10/450
6D4	Region Descriptor n, Word 5 (SMPU_RGD11_WRD5)	32	R/W	0000_0000h	20.3.11/451
700	Region Descriptor n, Word 0 (SMPU_RGD12_WRD0)	32	R/W	0000_0000h	20.3.6/446
704	Region Descriptor n, Word 1 (SMPU_RGD12_WRD1)	32	R/W	0000_000Fh	20.3.7/447
708	Region Descriptor n, Word 2 (SMPU_RD_12W2)	32	R/W	0000_0000h	20.3.8/447
70C	Region Descriptor n, Word 3 (SMPU_RGD12_WRD3)	32	R/W	0000_0000h	20.3.9/449
710	Region Descriptor n, Word 4 (SMPU_RGD12_WRD4)	32	R/W	0000_0000h	20.3.10/450
714	Region Descriptor n, Word 5 (SMPU_RGD12_WRD5)	32	R/W	0000_0000h	20.3.11/451
740	Region Descriptor n, Word 0 (SMPU_RGD13_WRD0)	32	R/W	0000_0000h	20.3.6/446
744	Region Descriptor n, Word 1 (SMPU_RGD13_WRD1)	32	R/W	0000_000Fh	20.3.7/447
748	Region Descriptor n, Word 2 (SMPU_RD_13W2)	32	R/W	0000_0000h	20.3.8/447
74C	Region Descriptor n, Word 3 (SMPU_RGD13_WRD3)	32	R/W	0000_0000h	20.3.9/449
750	Region Descriptor n, Word 4 (SMPU_RGD13_WRD4)	32	R/W	0000_0000h	20.3.10/450
754	Region Descriptor n, Word 5 (SMPU_RGD13_WRD5)	32	R/W	0000_0000h	20.3.11/451
780	Region Descriptor n, Word 0 (SMPU_RGD14_WRD0)	32	R/W	0000_0000h	20.3.6/446
784	Region Descriptor n, Word 1 (SMPU_RGD14_WRD1)	32	R/W	0000_000Fh	20.3.7/447
788	Region Descriptor n, Word 2 (SMPU_RD_14W2)	32	R/W	0000_0000h	20.3.8/447
78C	Region Descriptor n, Word 3 (SMPU_RGD14_WRD3)	32	R/W	0000_0000h	20.3.9/449
790	Region Descriptor n, Word 4 (SMPU_RGD14_WRD4)	32	R/W	0000_0000h	20.3.10/450
794	Region Descriptor n, Word 5 (SMPU_RGD14_WRD5)	32	R/W	0000_0000h	20.3.11/451
7C0	Region Descriptor n, Word 0 (SMPU_RGD15_WRD0)	32	R/W	0000_0000h	20.3.6/446
7C4	Region Descriptor n, Word 1 (SMPU_RGD15_WRD1)	32	R/W	0000_000Fh	20.3.7/447
7C8	Region Descriptor n, Word 2 (SMPU_RD_15W2)	32	R/W	0000_0000h	20.3.8/447
7CC	Region Descriptor n, Word 3 (SMPU_RGD15_WRD3)	32	R/W	0000_0000h	20.3.9/449
7D0	Region Descriptor n, Word 4 (SMPU_RGD15_WRD4)	32	R/W	0000_0000h	20.3.10/450
7D4	Region Descriptor n, Word 5 (SMPU_RGD15_WRD5)	32	R/W	0000_0000h	20.3.11/451

20.3.1 Control/Error Status Register 0 (SMPU_CES0)

CESR0 provides error status plus configuration information. A global SMPU enable/ disable bit is also included in this register.

Address: 0h base + 0h offset = 0h



SMPU_CES0 field descriptions

Field	Description
0–15 MERR	Master n error, where the bus master number matches the bit number Indicates a captured error in EARn and EDRn. A bit is set when the hardware detects an error and records the faulting address and attributes. It is cleared by writing one to it. If another error is captured at the exact same cycle as the write, the flag remains set. A find-first-one instruction (or equivalent) can detect the presence of a captured error. 0 No error has occurred for bus master n. 1 An error has occurred for bus master n.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 HRL	Hardware revision level Specifies the SMPU's hardware and definition revision level. It can be read by software to determine the functional definition of the module.
31 GVLD	Global Valid (global enable/disable for the SMPU)

Table continues on the next page...

SMPU_CES0 field descriptions (continued)

Field	Description
	0 SMPU is disabled. All accesses from all bus masters are allowed. 1 SMPU is enabled.

20.3.2 Control/Error Status Register 1 (SMPU_CES1)

CESR1 provides additional error status plus configuration information.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MEOVR																0												NRGD			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

SMPU_CES1 field descriptions

Field	Description
0–15 MEOVR	Master n error overrun, where the bus master number matches the bit number Each bit in this field signals that another SMPU error for bus master n has occurred before the previous error was processed. The details of the first error are recorded in the EARn and EDRn registers and no information on subsequent errors is recorded until the associated CESR0[MERR] flag is cleared. 0 No error overrun condition has been detected for bus master n. 1 An error overrun condition has been detected for bus master n.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 NRGD	Number of region descriptors Indicates the number of region descriptors implemented in the SMPU. The number of region descriptors is (4 x NRGD). 0001 4 region descriptors 0010 8 region descriptors 0011 12 region descriptors 0100 16 region descriptors 0101 20 region descriptors 0110 24 region descriptors

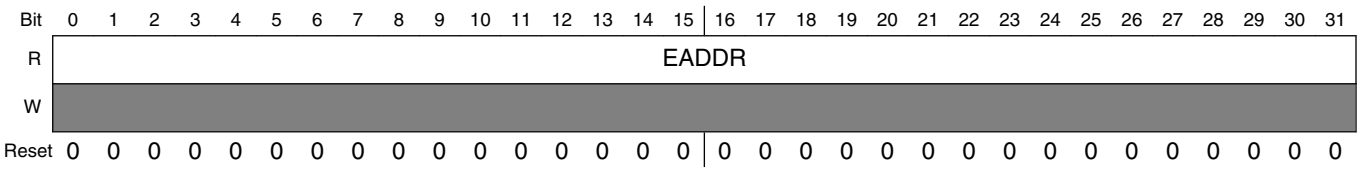
20.3.3 Error Address Register, Bus Master (SMPU_ERR_ADR*n*)

When the SMPU detects an access error on bus master n, the 32-bit reference address is captured in this read-only register and the corresponding bit in CESR0[MERR] set. Additional information about the faulting access is captured in the corresponding EDRn at the same time.

NOTE

The corresponding EARn and EDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1[MEOVR] field until the CESR0[MERR] flag associated with the original error is cleared (by writing a 1).

Address: 0h base + 100h offset + (16d × i), where i=0d to 15d



SMPU_ERR_ADR*n* field descriptions

Field	Description
0–31 EADDR	Error address Indicates the reference address from bus master n that generated the access error.

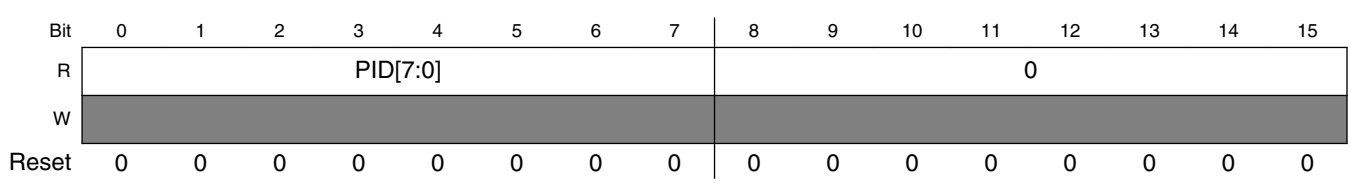
20.3.4 Error Attribute Detail Register, Bus Master (SMPU_ERR_ADTL*n*)

When the SMPU detects an access error associated with bus master n, 32 bits of error detail are captured in this read-only register and the corresponding bit in CESR0[MERR] isset. Information on the faulting address is captured in the corresponding EARn register at the same time.

NOTE

The corresponding EARn, EADRn, and EACDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1[MEOVR] Þeld until the original error is processed.

Address: 0h base + 104h offset + (16d × i), where i=0d to 15d



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						EATTR		ERW	EMN						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_ERR_ADTL*n* field descriptions

Field	Description
0–7 PID[7:0]	PID of offending access.
8–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 EATTR	Error attributes Indicates attribute information about the faulting reference. 00 User mode, instruction access 01 User mode, data access 10 Supervisor mode, instruction access 11 Supervisor mode, data access
24 ERW	Error read/write Indicates the access type of the faulting reference. 0 Read 1 Write
25–31 EMN	Error master number Indicates the logical bus master number of the faulting reference. This field is used to determine the bus master that generated the access error.

20.3.5 Error Attribute Control Detail Register, Bus Master (SMPU_ERR_ACDL*n*)

When the SMPU detects an access error associated with bus master n, 32 bits of error detail are captured in this read-only register and the corresponding bit in CESR0[MERR] is set. Information on the faulting address is captured in the corresponding EARn register at the same time.

NOTE

The corresponding EARn, EADRn, and EACDRn registers contain information on the original access error; subsequent errors associated with the given master are recorded as overruns in the CESR1[MEOVR] field until the original error is processed.

Address: 0h base + 108h offset + (16d × i), where i=0d to 15d

Bit	0–15	16–23	24–31
R	EACD	0	
W			
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0

SMPU_ERR_ACDL*n* field descriptions

Field	Description
0–23 EACD	Error access control detail Indicates the region descriptor with the access error, where the region descriptor number matches the bit number. If EDRn contains a captured error and EACD is all zeroes, an access did not hit in any region descriptor. If only a single EACD bit is set, the access error was caused by a single non-overlapping region descriptor. If two or more EACD bits are set, the access error was caused by an overlapping set of region descriptors. **NOTE:** When the address hits in a region but there is a PID miss (the PID is reflected in the Error Attribute Detail register), the field EACD = 0, if there is no other error. If there is another error, the EACD will reflect the region of the error (i.e., when it is hit in two regions)
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

20.3.6 Region Descriptor n, Word 0 (SMPU_RGD*n*_WRD0)

The first word of the region descriptor defines the byte start address of the memory region. Writes to this register clear the region descriptor's valid bit (RGDn_WORD5[VLD]).

Writes to ANY word of the descriptor clears the valid bit.

Address: 0h base + 400h offset + (64d × i), where i=0d to 15d

Bit	0–15	16–27	28–31
R	SRTADDR		Reserved
W			
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0

SMPU_RGD*n*_WRD0 field descriptions

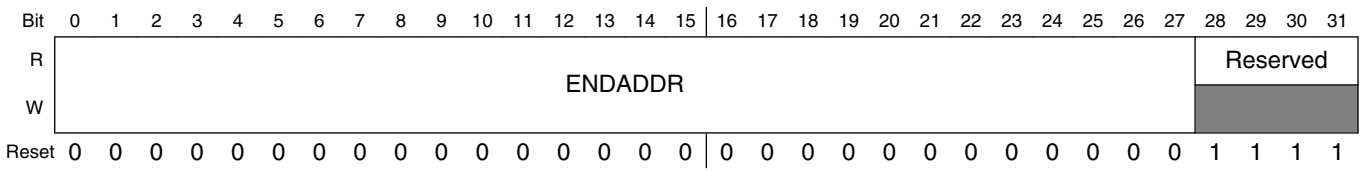
Field	Description
0–27 SRTADDR	Start address Defines the start address of the memory region.
28–31 Reserved	This field is reserved.

20.3.7 Region Descriptor n, Word 1 (SMPU_RGD*n*_WRD1)

The second word of the region descriptor defines the end address of the memory region. (RGDn_WORD5[VLD]).

Writes to ANY word of the descriptor clears the valid bit.

Address: 0h base + 404h offset + (64d × i), where i=0d to 15d



SMPU_RGD*n*_WRD1 field descriptions

Field	Description
0–27 ENDADDR	End address Defines the end address of the memory region. **NOTE:** 1. The SMPU module does not verify that a region's starting address (SMPU_RGDn_WRD0[SRTADDR]) is less than its ending address (SMPU_RGDn_WRD1[ENDADDR]). When a region's ending address is a lower address than the starting address, no error is generated and no hits can be generated for that region. 2. Both the starting address and the ending address are contained within a defined region. For example, the SRTADDR and ENDADDR default values (0000_0000h and 0000_000Fh, respectively) define a 16-byte region starting at address 0000_0000h and containing all bytes through address 0000_000Fh.
28–31 Reserved	This field is reserved.

20.3.8 Region Descriptor n, Word 2 (SMPU_RD_*n*W2)

RGD_WORD2 defines the access control rights of the memory region. The access control rights are defined by separate read and write permissions. For these fields, the bus master number refers to the *logical* bus master number.

For the access control rights, there are two flags:

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

Each field consists of the two flags, with (r) being in the more significant position. For example, M0P has (r) as bit 0 and (w) as bit 1.

- 00 - No Write or Read
- 01 - Write allowed, no Read
- 10 - Read allowed, no Write
- 11 - Both Read and Write allowed

Writes to this word clear the region descriptor's valid bit.

Address: 0h base + 408h offset + (64d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0P		M1P		M2P		M3P		M4P		M5P		M6P		M7P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8P		M9P		M10P		M11P		M12P		M13P		M14P		M15P	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RD_*n*W2 field descriptions

Field	Description
0–1 M0P	Bus master 0 permissions
2–3 M1P	Bus master 1 permissions
4–5 M2P	Bus master 2 permissions
6–7 M3P	Bus master 3 permissions
8–9 M4P	Bus master 4 permissions
10–11 M5P	Bus master 5 permissions
12–13 M6P	Bus master 6 permissions
14–15 M7P	Bus master 7 permissions
16–17 M8P	Bus master 8 permissions
18–19 M9P	Bus master 9 permissions
20–21 M10P	Bus master 10 permissions
22–23 M11P	Bus master 11 permissions
24–25 M12P	Bus master 12 permissions

Table continues on the next page...

SMPU_RD_*n*W2 field descriptions (continued)

Field	Description
26–27 M13P	Bus master 13 permissions
28–29 M14P	Bus master 14 permissions
30–31 M15P	Bus master 15 permissions

20.3.9 Region Descriptor n, Word 3 (SMPU_RGD*n*_WRD3)

The third word of the SMPU region descriptor contains the format select (FMT), three sets of access permissions flags used when FMT = 1, and the cache inhibit bit.

Writes to ANY word of the descriptor clears the valid bit.

Address: 0h base + 40Ch offset + (64d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												0		CI	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SMPU_RGD*n*_WRD3 field descriptions

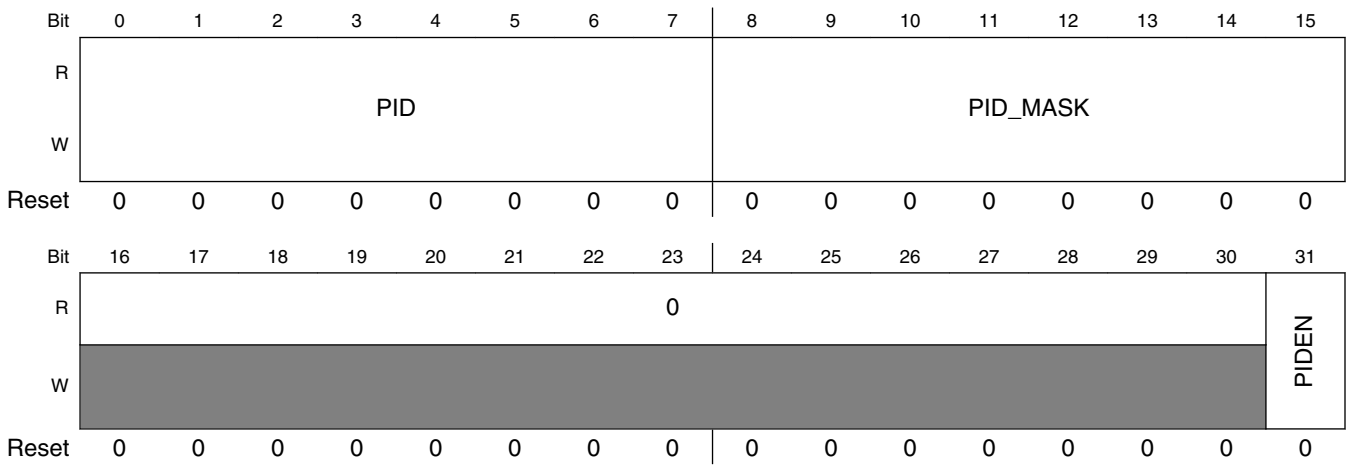
Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 CI	Cache Inhibit Defines the cacheability attribute of the memory region. This bit is returned to the bus master that initiated the memory reference. **NOTE:** An address range specified in an SMPU region descriptor for a cacheable space (that is, CI = 0) must be defined with a starting address aligned on a 0-modulo-32 byte address and with a multiple of the 32 byte cache line size factoring into the end address. For details on operation of CI attribute, please refer to CI note in Putting it all together and error terminations 0 References to this region can be cached. 1 References to this region cannot be cached.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

20.3.10 Region Descriptor n, Word 4 (SMPU_RGD*n*_WRD4)

The fifth word of the MPU region descriptor contains the optional process identifier and mask. If PIDEN is set, the PID and PID MASK of will be included in the hit evaluation calculation.

Writes to ANY word of the descriptor clears the valid bit.

Address: 0h base + 410h offset + (64d × i), where i=0d to 15d



SMPU_RGD*n*_WRD4 field descriptions

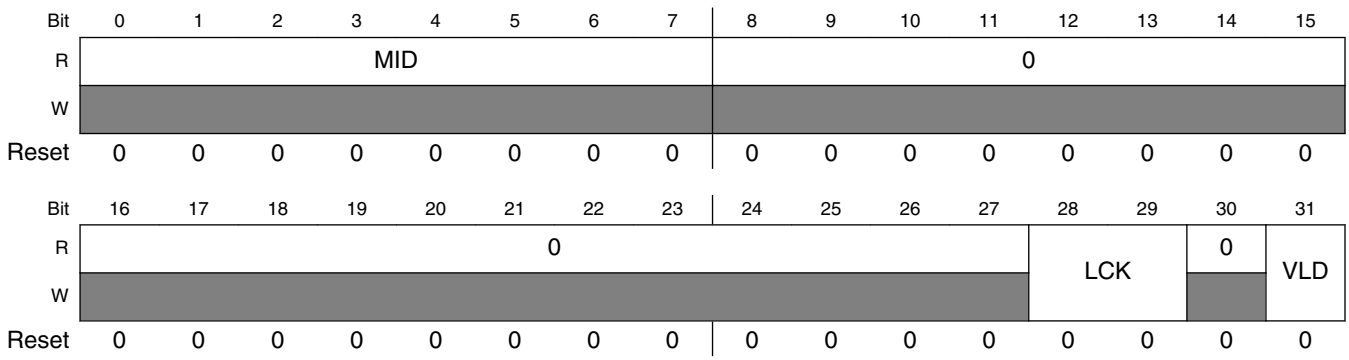
Field	Description
0–7 PID	Process Identifier This field specifies that the optional process identifier is to be included in the determination of whether the current access hits in the region descriptor. This field is combined with the PIDMASK and included in the region hit determination if SMPUx_RGDn_WORD3[PIDEN] is set.
8–15 PID_MASK	Process Identifier Mask This field provides a masking capability so that multiple process identifiers can be included as part of the region hit determination. If a bit in the PID_MASK is set, then the corresponding bit of the PID is ignored in the comparison. This field is combined with the PID and included in the region hit determination if SMPUx_RGDn_WORD3[PIDEN] is set. For more information on the handling of the PID and PID_MASK.
16–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 PIDEN	Region Descriptor PID Enable This flag specifies if the process identifier and mask (defined in MPU_RGDn_Word4) are to be included in the region hit evaluation 0 Do not include PID in the region hit evaluation 1 Include PID in the region hit evaluation

20.3.11 Region Descriptor n, Word 5 (SMPU_RGD*n*_WRD5)

The sixth word of the MPU region descriptor contains the master ID of the master who has locked the RGD as well as the Lock function and Valid bit. If LCK is set, the master ID of the master who has locked it will be stored here.

Writes to ANY word of the descriptor clears the valid bit.

Address: 0h base + 414h offset + (64d × i), where i=0d to 15d



SMPU_RGD*n*_WRD5 field descriptions

Field	Description
0–7 MID	Master ID of RGDn owner If this RGDn is locked through the setting of SMPUx_RGDn_WORD5[LCK], the MID field will reflect the Master ID of the current RGDn owner.
8–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–29 LCK	Region Descriptor Lock This bit allows a master to Lock a region descriptor. Once locked, only that master can modify the Region descriptor's contents. This includes all words of the RGD. **NOTE:** Setting LCK to 1x in an RGD locks all words of the RGD until system reset, the Valid bit of the RGD and the Global Valid bit of the SMPU have no effect. 00 Unlocked 01 Locked by the master wrote this register and LCK bit. Attempted writes by other masters are ignored with an error-free data transfer termination 10 This value is reserved for future enhancements and should not be used. 11 Attempted writes to any location in the region descriptor are ignored with an error-free data transfer termination.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 VLD	Valid Signals the region descriptor is valid. Any write to RGDn_WORD0–4 clears this bit. 0 Region descriptor is invalid 1 Region descriptor is valid

SMPU_RGD*n*_WRD5 field descriptions (continued)

Field	Description

20.4 Functional description

In this section, the functional operation of the SMPU is detailed, including the operation of the access evaluation macro and the handling of error-terminated bus cycles.

20.4.1 Access evaluation macro

The basic operation of the SMPU is performed in the access evaluation macro, a hardware structure replicated in the connection matrix. The following block diagram represents an individual macro.

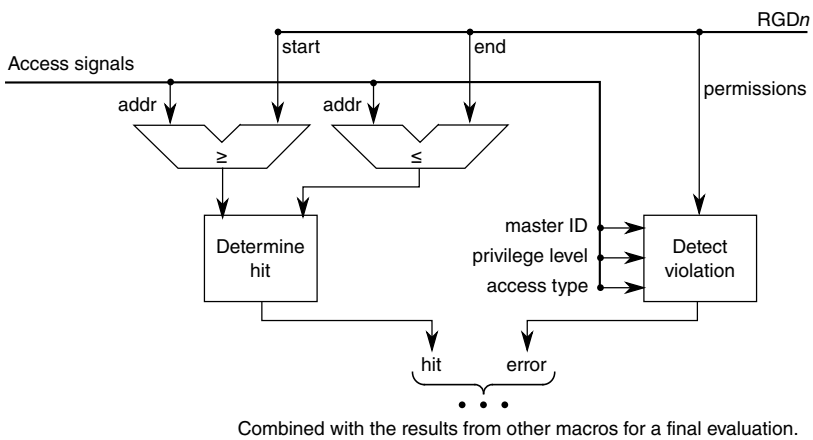


Figure 20-3. SMPU access evaluation macro

Each macro uses the information contained in the access signals (the address, the access type, and the master's ID and privilege level) and the contents of its corresponding region descriptor (RGD*n*) to perform two major functions: Hit determination and Violation determination. The individual hit and error results from each macro are then combined for a final evaluation; see Putting it all together and error terminations.

20.4.1.1 Hit determination

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

```
region_hit = ((addr[0:31] >= RGDn_Word0[SRTADDR]) & (addr[0:31] <= RGDn_Word1[ENDADDR])) &
RGDn_Word5[VLD]
```

where addr is the current reference address, RGD*n*_Word0[SRTADDR] and RGD*n*_Word1[ENDADDR] are the start and end addresses, and RGD*n*_Word5[VLD] is the valid bit.

NOTE

The SMPU does not verify that ENDADDR > SRTADDR.

NOTE

Region hit determination is based only on an address comparison of the first byte being accessed; that is, the SMPU does not check the size of the access to make sure it fits within an allowed region.

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

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

where rgdn.mxpe is the PID Enable of the RGD, the current_pid[*] is the selected process identifier from the current bus master, and rgdn.pid[*] and rgdn.pidmask[*] are the appropriate process identifier fields from the region descriptor n. For bus masters that do not output a process identifier, the SMPU forces the pid_hit term to be asserted. As shown in Figure 20-2, the access evaluation macro actually forms the logical complement (hit_b) of the combined region_hit and pid_hit boolean equations.

20.4.1.2 Violation determination

While the access evaluation macro is determining region hit, the logic is also evaluating if the current access is allowed by the permissions defined in the region descriptor. Using the master ID, a set of effective permissions is generated from the appropriate fields in the region descriptor. The protection violation logic then evaluates the access against the effective permissions using the specification shown below.

Table 20-2. Protection violation definition

Access type	Access permissions		Protection violation?
	r	w	
Instruction fetch read	0	—	Yes, no execute permission
	1	—	No, access is allowed
Data read	0	—	Yes, no read permission
	1	—	No, access is allowed
Data write	—	0	Yes, no write permission
	—	1	No, access is allowed

20.4.2 Putting it all together and error terminations

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

1. The access does not hit in any region descriptor.
2. The access hits in a single region descriptor and that region has a protection violation.
3. The access hits in multiple (overlapping) regions and all regions have protection violations.

As shown in the third condition, granting permission is a higher priority than denying access for overlapping regions. This approach is more flexible to system software in region descriptor assignments. For an example of the use of overlapping region descriptors, see Application information.

NOTE

For Cache-Inhibit (CI), it continues with least restrictive being higher priority. If there are overlapping regions, and at least one hitting region has CI=0, then it will not be Cache inhibited. It should be noted that CI only considers whether or not there is a hit, not whether there is a hit and permissions. For example, if two regions hit, one hit has permissions and has CI=1. Another region hit does not allow permissions, but has CI=0. This access would be granted, and it would not be cache inhibited.

NOTE

Aborted core accesses generate instruction storage (ISI) or data storage (DSI) interrupts (see the documentation related to the core for details).

20.5 Initialization information

At system startup, load the appropriate number of region descriptors, including setting RGD*n*_Word5[VLD]. Setting CESR0[GVLD] enables the module.

If the system requires that all the loaded region descriptors be enabled simultaneously, first ensure that the entire SMPU is disabled (CESR0[GVLD]=0). Next, load the appropriate region descriptors (RGDn), including the setting of the RGDn_Word5[VLD] bits. Finally, set CESR0[GVLD] to enable the entire module.

Note

A region descriptor must be set to allow access to the SMPU registers if further changes are needed.

20.6 Application information

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

- Creating a new memory region

  Load the appropriate region descriptor into an available RGD*n*, using four sequential 32-bit writes. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.

- Modifying a region descriptor

  Load the updates into the region descriptor using sequential 32-bit writes. Writing 1 to RGD*n*_WRD5[VLD] re-enables the region descriptor. The hardware assists in the maintenance of the valid bit, so if this approach is followed, there are no coherency issues with the multi-cycle descriptor writes.

- Removing a region descriptor

  Clearing RGD*n*_WRD5[VLD] deletes an existing region descriptor.

- Accessing the SMPU

Allocate a region descriptor to restrict SMPU access to supervisor mode from specific masters as appropriate.

- Detecting an access error

  The current bus cycle is terminated with an error response and EAR*n* and EDR*n* capture information on the faulting reference. The error-terminated bus cycle typically initiates an error response in the originating bus master. For example, a processor core may respond with a bus error exception, while a data movement bus master may respond with an error interrupt. The processor can retrieve the captured error address and detail information simply by reading E{A,D}R*n*. CESR0[MERR] signals which error registers contain captured fault data. Once the error information is retrieved, write a 1 to the appropriate CESR0[MERR] bit to clear it, thereby rearming the error capture logic associated with the EAR*n* and EDR*n* registers.

- Overlapping region descriptors

  Applying overlapping regions often reduces the number of descriptors required for a given set of access controls. In the overlapping memory space, the protection rights of the corresponding region descriptors are logically summed together (the boolean OR operator).

Chapter 21
Semaphores2 (SEMA42)

21.1 Introduction

The SEMA42 is a memory-mapped module that provides robust hardware support needed in multi-core systems for implementing semaphores and provides a simple mechanism to achieve "lock and unlock" operations via a single write access. The hardware semaphore module provides hardware-enforced gates as well as other useful system functions related to the gating mechanisms.

21.1.1 Features

- Supports 16 hardware-enforced gates in a multi-processor configuration
  - Gates appear as an 16-entry byte-size array with read and write accesses.
    - Processors lock gates by writing "processor_number+1" to the appropriate gate and must read back the gate value to verify the lock operation was successful.
    - Once locked, the gate is unlocked by a write of zeroes from the locking processor.
    - The number of implemented gates is specified by a hardware configuration define.
  - Each hardware gate appears as a 16-state, 4-bit state machine.
    - 16-state implementation

      if gate = 0x0, then state = unlocked
      if gate = 0x1, then state = locked by processor (master) 0
      if gate = 0x2, then state = locked by processor (master) 1
      …
      if gate = 0xF, then state = locked by processor (master) 14

- Uses the logical bus master number as a reference attribute plus the specified data patterns to validate all write operations.
- Once locked, the gate can (and must) be unlocked by a write of zeroes from the locking processor.

- Secure reset mechanisms are supported to clear the contents of individual gates, as well as a clear_all capability.

A simplified block diagram of the Semaphores module is shown in the following figure. In the diagram, the register blocks named gate0, gate1,…, gate (n-2), gate (n-1) include the finite state machines implementing the semaphore gates.

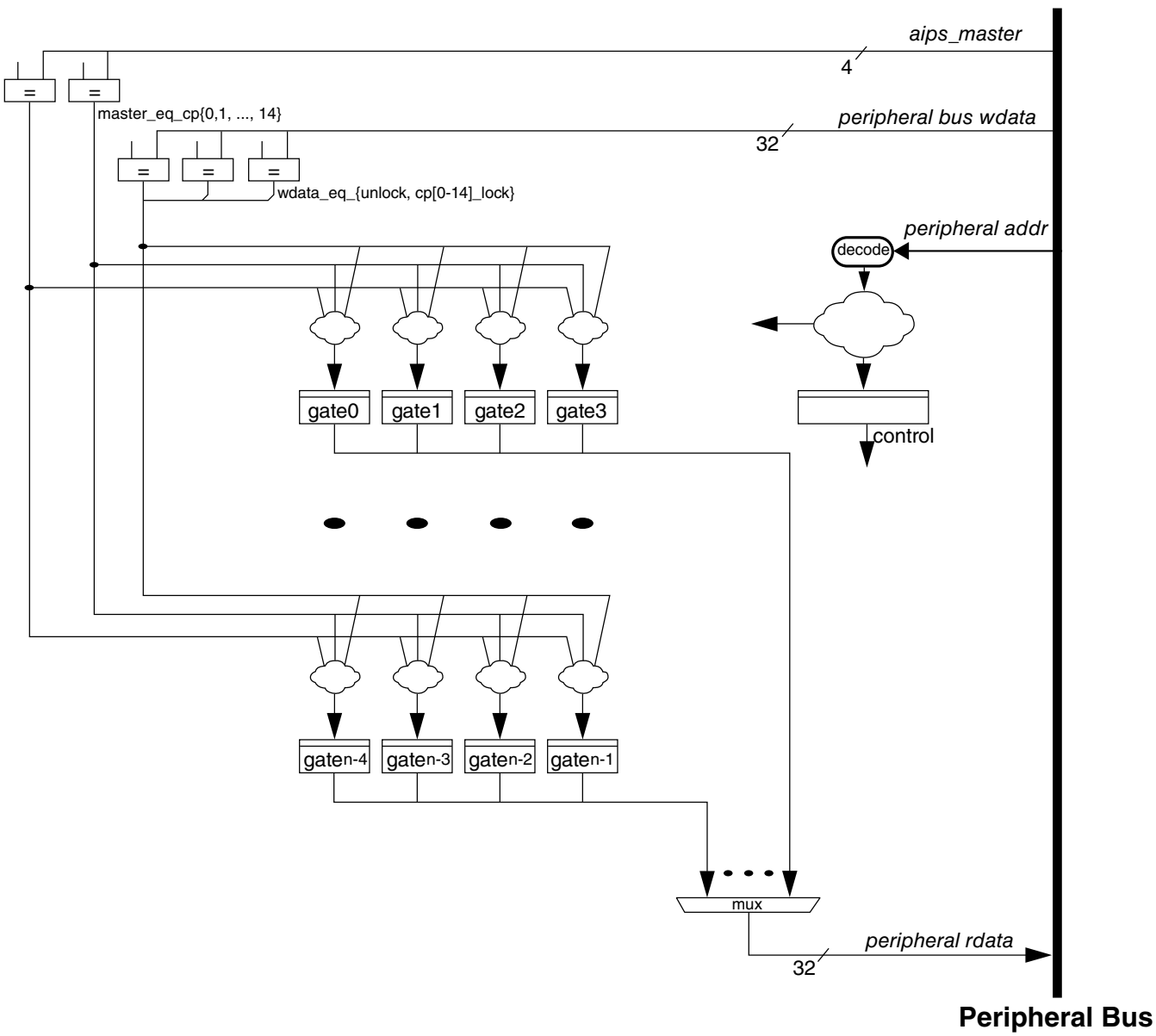


Figure 21-1. Semaphores2 block diagram

21.2 Memory map/register definition

Only Supervisor Mode accesses are allowed on these registers. User accesses will be terminated with an error.

SEMA42 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Gate Register (SEMA42_GATE0)	8	R/W	00h	21.2.1/459
1	Gate Register (SEMA42_GATE1)	8	R/W	00h	21.2.1/459
2	Gate Register (SEMA42_GATE2)	8	R/W	00h	21.2.1/459
3	Gate Register (SEMA42_GATE3)	8	R/W	00h	21.2.1/459
4	Gate Register (SEMA42_GATE4)	8	R/W	00h	21.2.1/459
5	Gate Register (SEMA42_GATE5)	8	R/W	00h	21.2.1/459
6	Gate Register (SEMA42_GATE6)	8	R/W	00h	21.2.1/459
7	Gate Register (SEMA42_GATE7)	8	R/W	00h	21.2.1/459
8	Gate Register (SEMA42_GATE8)	8	R/W	00h	21.2.1/459
9	Gate Register (SEMA42_GATE9)	8	R/W	00h	21.2.1/459
A	Gate Register (SEMA42_GATE10)	8	R/W	00h	21.2.1/459
B	Gate Register (SEMA42_GATE11)	8	R/W	00h	21.2.1/459
C	Gate Register (SEMA42_GATE12)	8	R/W	00h	21.2.1/459
D	Gate Register (SEMA42_GATE13)	8	R/W	00h	21.2.1/459
E	Gate Register (SEMA42_GATE14)	8	R/W	00h	21.2.1/459
F	Gate Register (SEMA42_GATE15)	8	R/W	00h	21.2.1/459
40	Reset Gate Write (SEMA42_RSTGT_W)	16	W	See section	21.2.2/460
40	Reset Gate Read (SEMA42_RSTGT_R)	16	R	0000h	21.2.3/462

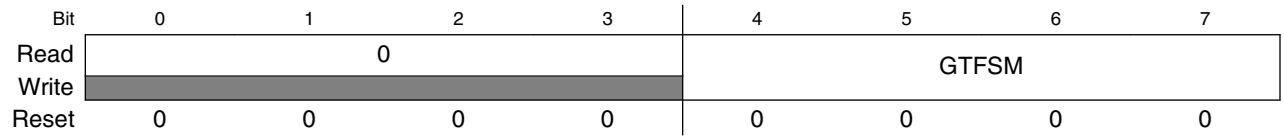
21.2.1 Gate Register (SEMA42_GATE*n*)

Each semaphore gate is implemented in a 4-bit finite state machine, right-justified in a byte data structure. The hardware uses the logical bus master number in conjunction with the data patterns to validate all attempted write operations. Only processor bus masters can modify the gate registers. Once locked, a gate can (and must) be opened (unlocked) by the locking processor core.

Multiple gate values can be read in a single access, but only a single gate can be updated via a write operation at a time. Attempted writes with a data value that is neither the unlock value (0x00) nor the appropriate lock value (processor_number + 1) are simply

treated as "no operation" and do not affect any gate state. Attempts to write multiple gates in a single aligned access with a size larger than an 8-bit (byte) reference generate an error termination and do not allow any gate state changes.

Address: 0h base + 0h offset + (1d × i), where i=0d to 15d



SEMA42_GATE*n* field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 GTFSM	Gate Finite State Machine. The state of the gate reflects the last processor that locked it, which can be useful during system debug. The hardware gate is maintained in a 16-state implementation, defined as: 0000 The gate is unlocked (free). 0001 The gate has been locked by processor 0. 0010 The gate has been locked by processor 1. 0011 The gate has been locked by processor 2. 0100 The gate has been locked by processor 3. 0101 The gate has been locked by processor 4. 0110 The gate has been locked by processor 5. 0111 The gate has been locked by processor 6. 1000 The gate has been locked by processor 7. 1001 The gate has been locked by processor 8. 1010 The gate has been locked by processor 9. 1011 The gate has been locked by processor 10. 1100 The gate has been locked by processor 11. 1101 The gate has been locked by processor 12. 1110 The gate has been locked by processor 13. 1111 The gate has been locked by processor 14.

21.2.2 Reset Gate Write (SEMA42_RSTGT_W)

NOTE

This section and the following section, Reset Gate Read (SEMA42_RSTGT_R), describe the same register. This section describes the general operation of the register and also shows how the register fields appear when the register is written. Reset Gate Read (SEMA42_RSTGT_R) shows how the register fields appear when the register is read.

Although the intent of the hardware gate implementation specifies a protocol where the locking domain must unlock the gate, it is recognized that system operation may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset.

To support this special gate reset requirement, the Semaphores module implements a "secure" reset mechanism that allows a hardware gate (or all the gates) to be initialized by following a specific dual-write access pattern. Using a technique similar to that required for the servicing of a software watchdog timer, the secure gate reset requires two consecutive writes with predefined data patterns from the same domain to force the clearing of the specified gate(s). The required access pattern is:

1. A domain performs a 16-bit write to the SEMA42_RSTGT memory location. The most significant byte (SEMA42_RSTGT[RSTGDP]) must be E2h; the least significant byte is a "don't_care" for this reference.
2. The same domain then performs a second 16-bit write to the SEMA42_RSTGT location. For this write, the upper byte (SEMA42_RSTGT[RSTGDP]) is the logical complement of the first data pattern (1Dh) and the lower byte (SEMA42_RSTGT[RSTGTN]) specifies the gate(s) to be reset. This gate field can specify a single gate be cleared, or that all gates are to be cleared. If the same domain writes incorrect data on the second access or another domain performs the second write access, the special gate reset sequence is aborted and no error signal is asserted.
3. Reads of the SEMA42_RSTGT location return information on the 2-bit state machine (SEMA42_RSTGT[RSTGSM]) that implements this function, the domain performing the reset (SEMA42_RSTGT[RSTGMS]), and the gate number(s) last cleared (SEMA42_RSTGT[RSTGTN]). Reads of the SEMA42_RSTGT register do not affect the secure reset finite state machine in any manner.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read																
Write	RSTGDP								RSTGTN							
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value is not applicable for writes.

SEMA42_RSTGT_W field descriptions

Field	Description
0–7 RSTGDP	Reset Gate Data Pattern. This write-only field is accessed with the specified data patterns on the two consecutive writes to enable the gate reset mechanism. For the first write, RSTGDP = E2h while the second write requires RSTGDP = 1Dh.

Table continues on the next page...

SEMA42_RSTGT_W field descriptions (continued)

Field	Description
8–15 RSTGTN	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates.

21.2.3 Reset Gate Read (SEMA42_RSTGT_R)

This section describes how the Reset Gate register fields appear when the register is read. See Reset Gate Write (SEMA42_RSTGT_W) for the description of the register's overall operation and how the register fields appear when the register is written.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	ROZ		RSTGSM		RSTGMS				RSTGTN							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SEMA42_RSTGT_R field descriptions

Field	Description
0–1 ROZ	This field always returns the value 0 when read.
2–3 RSTGSM	Reset Gate Finite State Machine. Reads of the SEMA42_RSTGT register return the encoded state machine value. Note the RSTGSM = 10 state is valid for only a single machine cycle, so it is impossible for a read to return this value. The reset state machine is maintained in a 2-bit, 3-state implementation, defined as: 00 Idle, waiting for the first data pattern write. 01 Waiting for the second data pattern write. 10 The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state. The "01" state persists for only one clock cycle. Software cannot observe this state. 11 This state encoding is never used and therefore reserved.
4–7 RSTGMS	Reset Gate Bus Master. This 4-bit read-only field records the logical number of the bus master performing the gate reset function. The reset function requires that the two consecutive writes to this register must be initiated by the same bus master to succeed. This field is updated each time a write to this register occurs. The association between system bus master port numbers, the associated bus master device, and the logical processor number is SoC-specific. Consult the device reference manual for this information.
8–15 RSTGTN	Reset Gate Number. This 8-bit field specifies the specific hardware gate to be reset. This field is updated by the second write. If RSTGTN < 64, then reset the single gate defined by RSTGTN, else reset all the gates.

21.3 Functional description

Functional operation of the Semaphores module and specific details of the state machines of the SEMA42_GATEn registers are provided as follows.

As described previously, each of the SEMA42_GATEn registers implements a 4-bit, 16-state machine. A *simplified* diagram of the state transitions for each gate is shown in Figure 21-2.

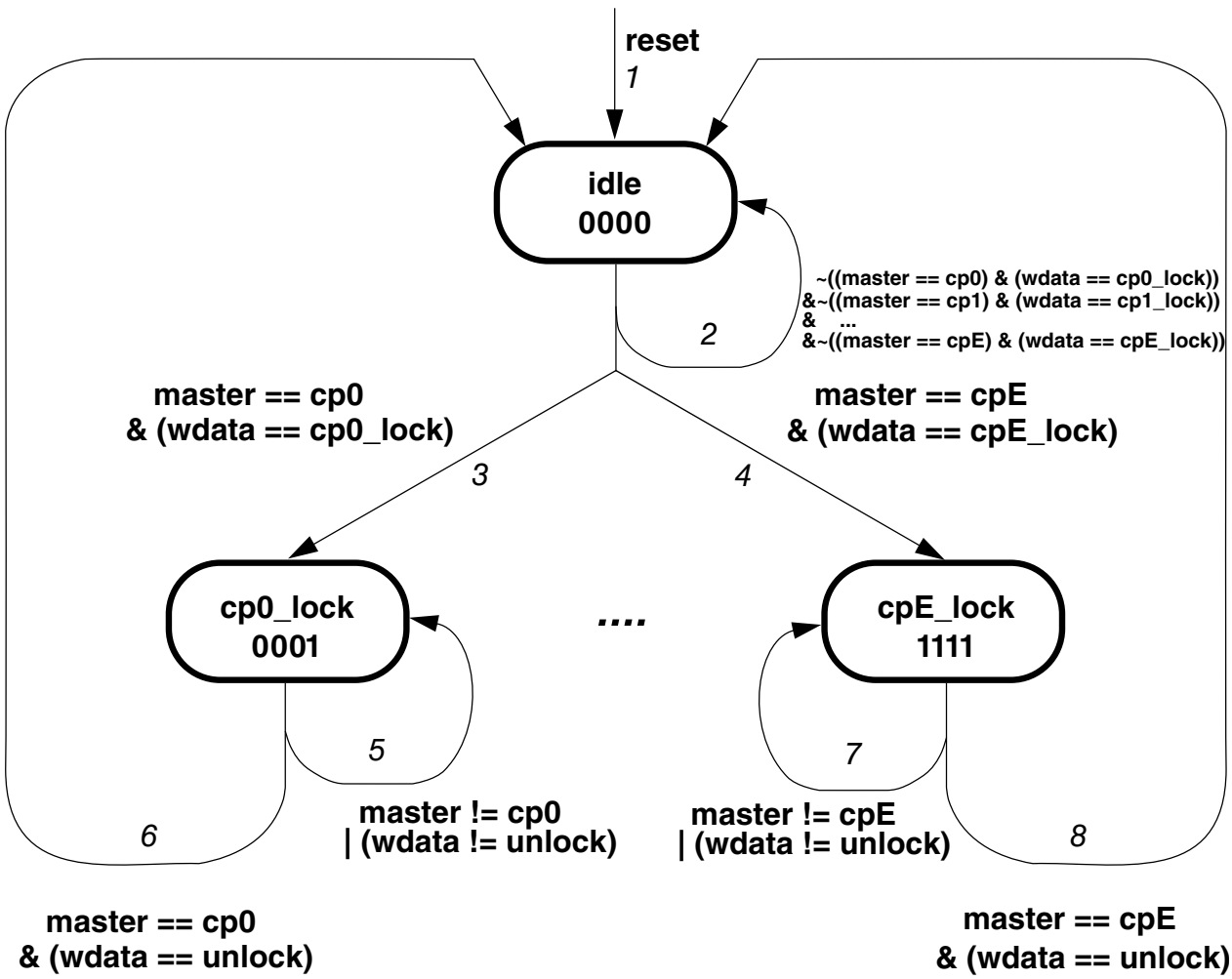


Figure 21-2. SEMA42_GATEn state machine

The bus master number is used to identify core processor *X* (cp*X*) where the notation cpE is used to represent core processor 14 (= 0xE). The platform passes the AHB bus master number through the peripheral bridge controller and drives a signal to the Semaphores module.

The state transitions for SEMA42_GATEn are defined in Table 21-1.

Table 21-1. SEMA42_GATEn state transitions

Current State	Next State	Transition	Description
–	idle	1	Any reset, whether a system reset or a software-initiated gate reset, unconditionally forces the gate into the idle state.
idle	idle	2	Unless a write of the appropriate lock value from the corresponding processor occurs, the gate remains in the idle state.
idle	cp0_lock	3	When a write of the "cp0_lock" data value is initiated by processor 0, the gate transitions into the cp0_lock state.
idle	cpE_lock	4	When a write of the "cpE_lock" value is initiated by processor 0xE, the gate transitions into the cpE_lock state.
cp0_lock	cp0_lock	5	Once in this state, the gate remains here if any attempted write is not from cp0 with the unlock data value.
cp0_lock	idle	6	The gate returns to the idle (unlocked) state once a write from cp0 with the unlock data value occurs.
cpE_lock	cpE_lock	7	Once in this state, the gate remains here if any attempted write is not from cpE with the unlock data value.
cpE_lock	idle	8	The gate returns to the idle (unlocked) state once a write from cpE with the unlock data value occurs.

The following gate data values are used:

- The lock data value is (processor number + 1) where the processor number and the platform bus master number are the same.
- The unlock data value is 0x00.

Chapter 22
Platform Configuration Module (PCM)

22.1 Memory map and register definition

The Platform Configuration Module (PCM) contains miscellaneous configuration registers for the device. Currently, the configuration registers are related to the operation of the intelligent bus bridging gasket. The module is mapped to PBRIDGE on-platform slot 10 with a base address of 0xFC02_8000. This PCM drives ports on the intelligent bridging gaskets and Clock Gating Module

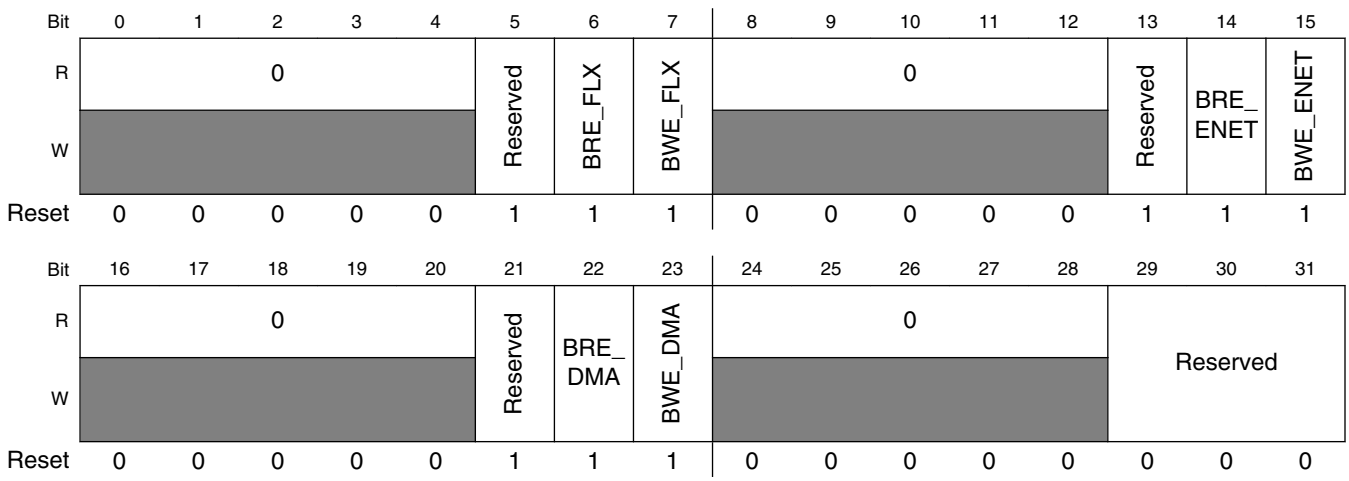
PCM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Bus Bridge Configuration Register 0 (PCM_IAHB_BE0)	32	R/W	0707_0700h	22.1.1/466
4	Bus Bridge Configuration Register 1 (PCM_IAHB_BE1)	32	R/W	0707_0707h	22.1.2/468
8	IPS Clock Gating Module Enable 0 Register (PCM_IPS_CGM_EN0)	32	R/W	0011_1111h	22.1.3/470

22.1.1 Bus Bridge Configuration Register 0 (PCM_IAHB_BE0)

This register must be accessed in supervisor mode.

Address: 0h base + 0h offset = 0h



PCM_IAHB_BE0 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 Reserved	This field is reserved.
6 BRE_FLX	Burst read enable (BRE) FLX This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket 1 Burst reads are optimized for best system performance
7 BWE_FLX	Burst Write Enable This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved.
14 BRE_ENET	Burst read enable (BRE) ENET This bit controls the bus gasket's handling of burst read transactions

Table continues on the next page...

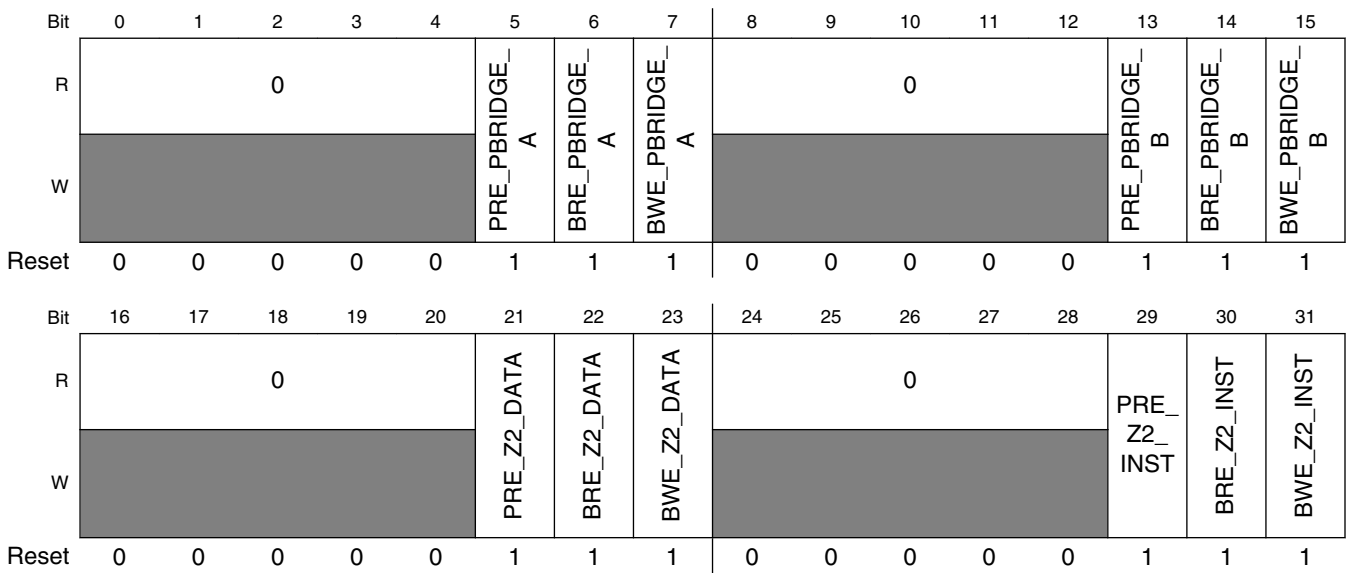
PCM_IAHB_BE0 field descriptions (continued)

Field	Description
	0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_ENET	Burst Write Enable ENET This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved.
22 BRE_DMA	Burst read enable (BRE) DMA This bit controls the bus gasket's handling of burst read transactions 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance
23 BWE_DMA	Burst write enable (BWE) DMA This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as “imprecise” such that an error response on any beat of the burst is reported on the last beat.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 Reserved	This field is reserved.

22.1.2 Bus Bridge Configuration Register 1 (PCM_IAHB_BE1)

This register must be accessed in supervisor mode.

Address: 0h base + 4h offset = 4h



PCM_IAHB_BE1 field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 PRE_PBRIDGE_A	Pending read enable (PRE) PBRIDGE_A This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled 1 Pending reads are enabled
6 BRE_PBRIDGE_A	Burst read enable (BRE) PBRIDGE_A This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket 1 Burst reads are optimized for best system performance
7 BWE_PBRIDGE_A	Burst Write Enable PBRIDGE_A This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.

Table continues on the next page...

PCM_IAHB_BE1 field descriptions (continued)

Field	Description
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 PRE_PBRIDGE_B	Pending Read Enable PBRIDGE_B This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled. 1 Pending reads are enabled.
14 BRE_PBRIDGE_B	Burst read enable (BRE) PBRIDGE_B This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance.
15 BWE_PBRIDGE_B	Burst Write Enable PBRIDGE_B This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 PRE_Z2_DATA	Pending read enable (PRE) Z2_DATA This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled 1 Pending reads are enabled
22 BRE_Z2_DATA	Burst read enable (BRE) Z2_DATA This bit controls the bus gasket's handling of burst read transactions 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance
23 BWE_Z2_DATA	Burst write enable (BWE) Z2_DATA This bit controls the bus gasket's handling of burst write transactions. 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 PRE_Z2_INST	Pending read enable (PRE) Z2_INST This bit controls the bus gasket's handling of pending read transactions. 0 Pending reads are disabled 1 Pending reads are enabled

Table continues on the next page...

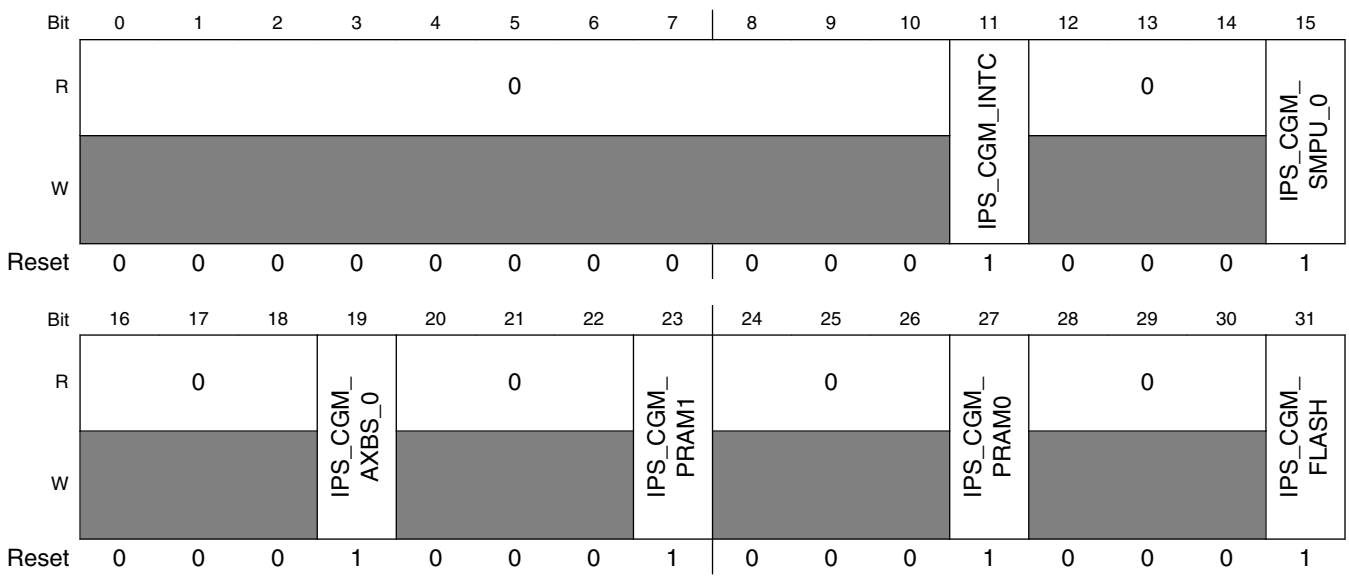
PCM_IAHB_BE1 field descriptions (continued)

Field	Description
30 BRE_Z2_INST	Burst read enable (BRE) Z2_INST This bit controls the bus gasket's handling of burst read transactions. 0 Burst reads are converted into a series of single transactions on the slave side of the gasket. 1 Burst reads are optimized for best system performance
31 BWE_Z2_INST	Burst write enable (BWE) Z2_INST This bit controls the bus gasket's handling of burst write transactions 0 Burst writes are converted into a series of single transactions on the slave side of the gasket. 1 Burst writes are optimized for best system performance. Note this setting treats writes as "imprecise" such that an error response on any beat of the burst is reported on the last beat

22.1.3 IPS Clock Gating Module Enable 0 Register (PCM_IPS_CGM_EN0)

This register must be accessed in supervisor mode.

Address: 0h base + 8h offset = 8h



PCM_IPS_CGM_EN0 field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PCM_IPS_CGM_EN0 field descriptions (continued)

Field	Description
11 IPS_CGM_INTC	IPS clock gating enable for INTC 1 Enable clock gating, one IPS clock delay is introduced between master and slave. 0 Disable clock gating
12–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 IPS_CGM_ SMPU_0	IPS clock gating enable for SMPU_0 1 Enable clock gating, one IPS clock delay is introduced between master and slave 0 Disable clock gating
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 IPS_CGM_ AXBS_0	IPS clock gating enable for AXBS_0 1 Enable clock gating, one IPS clock delay is introduced between master and slave 0 Disable clock gating
20–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 IPS_CGM_ PRAM1	IPS clock gating enable for PRAM1 1 Enable clock gating, one IPS clock delay is introduced between master and slave 0 Disable clock gating
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 IPS_CGM_ PRAM0	IPS clock gating enable for PRAM0 1 Enable clock gating, one IPS clock delay is introduced between master and slave 0 Disable clock gating
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 IPS_CGM_ FLASH	IPS clock gating enable for PFlash Controller 1 Enable clock gating, one IPS clock delay is introduced between master and slave 0 Disable clock gating

Chapter 23
Interrupt Controller (INTC)

23.1 Chip-specific INTC information

23.1.1 INTC configuration

23.1.1.1 INTC configurable parameters

The INTC on this chip supports 2 processors (0, 1) and 16 level priorities of each interrupt.

23.1.2 INTC interrupt sources

The following table defines the interrupt sources for the INTC on this chip. All IRQ numbers not specifically listed are not used.

NOTE

All interrupts which are masked in INTC should also be marked as disabled in the respective modules.

Table 23-1. INTC Vector Table

Vector number	Vector offset	Source module	Source description
0	1000	Software setable flag 0	SSCIR0[CLR0]
1	1004	Software setable flag 1	SSCIR0[CLR1]
2	1008	Software setable flag 2	SSCIR0[CLR2]
3	100C	Software setable flag 3	SSCIR0[CLR3]
4	1010	Software setable flag 4	SSCIR0[CLR4]
5	1014	Software setable flag 5	SSCIR0[CLR5]
6	1018	Software setable flag 6	SSCIR0[CLR6]

Table continues on the next page...

Table 23-1. INTC Vector Table (continued)

Vector number	Vector offset	Source module	Source description
7	101C	Software setable flag 7	SSCIR0[CLR7]
8	1020	Software setable flag 8	SSCIR0[CLR8]
9	1024	Software setable flag 9	SSCIR0[CLR9]
10	1028	Software setable flag 10	SSCIR0[CLR10]
11	102C	Software setable flag 11	SSCIR0[CLR11]
12	1030	Software setable flag 12	SSCIR0[CLR12]
13	1034	Software setable flag 13	SSCIR0[CLR13]
14	1038	Software setable flag 14	SSCIR0[CLR14]
15	103C	Software setable flag 15	SSCIR0[CLR15]
16	1040	Software setable flag 16	SSCIR0[CLR16]
17	1044	Software setable flag 17	SSCIR0[CLR17]
18	1048	Software setable flag 18	SSCIR0[CLR18]
19	104C	Software setable flag 19	SSCIR0[CLR19]
20	1050	Software setable flag 20	SSCIR0[CLR20]
21	1054	Software setable flag 21	SSCIR0[CLR21]
22	1058	Software setable flag 22	SSCIR0[CLR22]
23	105C	Software setable flag 23	SSCIR0[CLR23]
32	1080	Platform watchdog timer0	SWT_0_IR[TIF]
33	1084	Platform watchdog timer1	SWT_1_IR[TIF]
36	1090	On-Platform periodic timer 0_0 (STM)	STM_0_CIR0[CIF]
37	1094	On-Platform periodic timer 0_1 (STM)	STM_0_CIR1[CIF]
38	1098	On-Platform periodic timer 0_2 (STM)	STM_0_CIR2[CIF]
39	109A	On-Platform periodic timer 0_3 (STM)	STM_0_CIR3[CIF]
40	10A0	On-Platform periodic timer 1_0 (STM)	STM_1_CIR0[CIF]
41	10A4	On-Platform periodic timer 1_1 (STM)	STM_1_CIR1[CIF]
42	10A8	On-Platform periodic timer 1_2 (STM)	STM_1_CIR2[CIF]
43	10AC	On-Platform periodic timer 1_3 (STM)	STM_1_CIR3[CIF]
52	10D0	eDMA Combined Error	eDMA Channel Error Flags 63-0
53	10D4	eDMA Channel 0	DMA_INTL[INT0]
54	10D8	eDMA Channel 1	DMA_INTL[INT1]
55	10DC	eDMA Channel 2	DMA_INTL[INT2]
56	10E0	eDMA Channel 3	DMA_INTL[INT3]
57	10E4	eDMA Channel 4	DMA_INTL[INT4]
58	10E8	eDMA Channel 5	DMA_INTL[INT5]

Table continues on the next page...

Table 23-1. INTC Vector Table (continued)

Vector number	Vector offset	Source module	Source description
59	10EC	eDMA Channel 6	DMA_INTL[INT6]
60	10F0	eDMA Channel 7	DMA_INTL[INT7]
61	10F4	eDMA Channel 8	DMA_INTL[INT8]
62	10F8	eDMA Channel 9	DMA_INTL[INT9]
63	10FC	eDMA Channel 10	DMA_INTL[INT10]
64	1100	eDMA Channel 11	DMA_INTL[INT11]
65	1104	eDMA Channel 12	DMA_INTL[INT12]
66	1108	eDMA Channel 13	DMA_INTL[INT13]
67	110C	eDMA Channel 14	DMA_INTL[INT14]
68	1110	eDMA Channel 15	DMA_INTL[INT15]
69	1114	eDMA Channel 16	DMA_INTL[INT16]
70	1118	eDMA Channel 17	DMA_INTL[INT17]
71	111C	eDMA Channel 18	DMA_INTL[INT18]
72	1120	eDMA Channel 19	DMA_INTL[INT19]
73	1124	eDMA Channel 20	DMA_INTL[INT20]
74	1128	eDMA Channel 21	DMA_INTL[INT21]
75	112C	eDMA Channel 22	DMA_INTL[INT22]
76	1130	eDMA Channel 23	DMA_INTL[INT23]
77	1134	eDMA Channel 24	DMA_INTL[INT24]
78	1138	eDMA Channel 25	DMA_INTL[INT25]
79	113C	eDMA Channel 26	DMA_INTL[INT26]
80	1140	eDMA Channel 27	DMA_INTL[INT27]
81	1144	eDMA Channel 28	DMA_INTL[INT28]
82	1148	eDMA Channel 29	DMA_INTL[INT29]
83	114C	eDMA Channel 30	DMA_INTL[INT30]
84	1150	eDMA Channel 31	DMA_INTL[INT31]
185	12E4	Flash controller Prog/Erase/ Suspend IRQ_0	MCR[DONE]
210	1348	ENET Interrupt Group0	ENET_0
211	1370	ENET Interrupt Group1	ENET_1
212	1374	ENET Interrupt Group2	ENET_2
213	1378	ENET Interrupt Group3	ENET_3
214	137C	ENET Interrupt Group4	ENET_4
215	1380	ENET Interrupt Group5	ENET_5
216	1384	ENET Interrupt Group6	ENET_6
217	1364	ENET Interrupt Group7	ENET_7
218	1368	ENET Interrupt Group8	ENET_8
224	1380	Real Time Counter (RTC)	RTC
225	1384	Autonomous Periodic Interrupt (API)	API
226	1388	Periodic Interrupt Timer (PIT0)	PIT_1_TFLG0[TIF]

Table continues on the next page...

Table 23-1. INTC Vector Table (continued)

Vector number	Vector offset	Source module	Source description
227	138C	Periodic Interrupt Timer (PIT1)	PIT_1_TFLG1[TIF]
228	1390	Periodic Interrupt Timer (PIT2)	PIT_1_TFLG2[TIF]
229	1394	Periodic Interrupt Timer (PIT3)	PIT_1_TFLG3[TIF]
230	1398	Periodic Interrupt Timer (PIT4)	PIT_1_TFLG4[TIF]
231	139C	Periodic Interrupt Timer (PIT5)	PIT_1_TFLG5[TIF]
232	13A0	Periodic Interrupt Timer (PIT6)	PIT_1_TFLG6[TIF]
233	13A4	Periodic Interrupt Timer (PIT7)	PIT_1_TFLG7[TIF]
234	13A8	Periodic Interrupt Timer (PIT8)	PIT_1_TFLG8[TIF]
235	13AC	Periodic Interrupt Timer (PIT9)	PIT_1_TFLG9[TIF]
236	13B0	Periodic Interrupt Timer (PIT10)	PIT_1_TFLG10[TIF]
237	13B4	Periodic Interrupt Timer (PIT11)	PIT_1_TFLG11[TIF]
238	13B8	Periodic Interrupt Timer (PIT12)	PIT_1_TFLG12[TIF]
239	13BC	Periodic Interrupt Timer (PIT13)	PIT_1_TFLG13[TIF]
240	13C0	Periodic Interrupt Timer (PIT14)	PIT_1_TFLG14[TIF]
241	13C4	Periodic Interrupt Timer (PIT15)	PIT_1_TFLG15[TIF]
242	13C8	PIT_RTI	PIT_RTI
243	13CC	SIUL	EIRQ [0-7] - SIUL Combined External Interrupt
244	13D0	SIUL	EIRQ [8-15] - SIUL Combined External Interrupt
245	13D4	SIUL	EIRQ [16-23] - SIUL Combined External Interrupt
246	13D8	SIUL	EIRQ [24-31] - SIUL Combined External Interrupt
250	13E8	LPU_CTL	LPU_ICR[NEM] | LPU_ICR[MRIG]
251	13EC	MC _ME	ME_IS[I_SAFE]
252	13F0	MC _ME	ME_IS[I_MTC]
253	13F4	MC _ME	ME_IS[I_IMODE]
254	13F8	MC _ME	ME_IS[I_ICONF]
255	13FC	MC_RGM	MC_RGM Functional and destructive reset alternate event interrupt
257	1404	FXOSC Counter	FXOSC Counter
258	1408	SXOSC Counter	SXOSC Counter
259	140C	DSPI0_0	DSPI_0_SR[TFUF] | DSPI_0_SR[RFOF]| DSPI_0_SR[TFIWF]
260	1410	DSPI0_1	DSPI_0_SR[EOQF]
261	1414	DSPI0_2	DSPI_0_SR[TFFF]
262	1418	DSPI0_3	DSPI_0_SR[TCF]
263	141C	DSPI0_4	DSPI_0_SR[RFDF]
264	1420	DSPI0_5	DSPI_0_SR[SPITCF] | DSPI_0_SR[CMD_TCF]
265	1424	DSPI0_6	DSPI_0_SR[DSITCF] | DSPI_0_SR[CMDFFF]
266	1428	DSPI0_7	DSPI_0_SR[SPEF] | DSPI_0_SR[DPEF]
267	142C	DSPI0_8	DSPI_0_SR[DDIF]

Table continues on the next page...

Table 23-1. INTC Vector Table (continued)

Vector number	Vector offset	Source module	Source description
268	1430	DSPI1_0	DSPI_1_SR[TFUF] | DSPI_1_SR[RFOF]| DSPI_1_SR[TFIWF]
269	1434	DSPI1_1	DSPI_1_SR[EOQF]
270	1438	DSPI1_2	DSPI_1_SR[TFFF]
271	143C	DSPI1_3	DSPI_1_SR[TCF]
272	1440	DSPI1_4	DSPI_1_SR[RFDF]
273	1444	DSPI1_5	DSPI_1_SR[SPITCF] | DSPI_1_SR[CMD_TCF]
274	1448	DSPI1_6	DSPI_1_SR[DSITCF] | DSPI_1_SR[CMDFFF]
275	144C	DSPI1_7	DSPI_1_SR[SPEF] | DSPI_1_SR[DPEF]
276	1450	DSPI1_8	DSPI_1_SR[DDIF]
277	1454	DSPI2_0	DSPI_2_SR[TFUF] | DSPI_2_SR[RFOF]| DSPI_2_SR[TFIWF]
278	1458	DSPI2_1	DSPI_2_SR[EOQF]
279	145C	DSPI2_2	DSPI_2_SR[TFFF]
280	1460	DSPI2_3	DSPI_2_SR[TCF]
281	1464	DSPI2_4	DSPI_2_SR[RFDF]
282	1468	DSPI2_5	DSPI_2_SR[SPITCF] | DSPI_2_SR[CMD_TCF]
283	146C	DSPI2_6	DSPI_2_SR[DSITCF] | DSPI_2_SR[CMDFFF]
284	1470	DSPI2_7	DSPI_2_SR[SPEF] | DSPI_2_SR[DPEF]
285	1474	DSPI2_8	DSPI_2_SR[DDIF]
286	1478	DSPI3_0	DSPI_3_SR[TFUF] | DSPI_3_SR[RFOF]| DSPI_3_SR[TFIWF]
287	147C	DSPI3_1	DSPI_3_SR[EOQF]
288	1480	DSPI3_2	DSPI_3_SR[TFFF]
289	1484	DSPI3_3	DSPI_3_SR[TCF]
290	1488	DSPI3_4	DSPI_3_SR[RFDF]
291	148C	DSPI3_5	DSPI_3_SR[SPITCF] | DSPI_3_SR[CMD_TCF]
292	1490	DSPI3_6	DSPI_3_SR[DSITCF] | DSPI_3_SR[CMDFFF]
293	1494	DSPI3_7	DSPI_3_SR[SPEF] | DSPI_3_SR[DPEF]
294	1498	DSPI3_8	DSPI_3_SR[DDIF]
295	149C	SPI0_0	SPI_0_SR[TFUF] | SPI_0_SR[RFOF]| SPI_0_SR[TFIWF]
296	14A0	SPI0_1	SPI_0_SR[EOQF]
297	14A4	SPI0_2	SPI_0_SR[TFFF]
298	14A8	SPI0_3	SPI_0_SR[TCF]
299	14AC	SPI0_4	SPI_0_SR[RFDF]
300	14B0	SPI0_5	SPI_0_SR[SPITCF] | SPI_0_SR[CMD_TCF]
301	14B4	SPI0_6	SPI_0_SR[DSITCF] | SPI_0_SR[CMDFFF]
302	14B8	SPI0_7	SPI_0_SR[SPEF] | SPI_0_SR[DPEF]
304	14C0	SPI1_0	SPI_1_SR[TFUF] | SPI_1_SR[RFOF]| SPI_1_SR[TFIWF]

Table continues on the next page...

Table 23-1. INTC Vector Table (continued)

Vector number	Vector offset	Source module	Source description
305	14C4	SPI1_1	SPI_1_SR[EOQF]
306	14C8	SPI1_2	SPI_1_SR[TFFF]
307	14CC	SPI1_3	SPI_1_SR[TCF]
308	14D0	SPI1_4	SPI_1_SR[RFDF]
309	14D4	SPI1_5	SPI_1_SR[SPITCF] | SPI_1_SR[CMD_TCF]
310	14D8	SPI1_6	SPI_1_SR[DSITCF] | SPI_1_SR[CMDFFF]
311	14DC	SPI1_7	SPI_1_SR[SPEF] | SPI_1_SR[DPEF]
313	14E4	SPI2_0	SPI_2_SR[TFUF] | SPI_2_SR[RFOF]| SPI_2_SR[TFIWF]
314	14E8	SPI2_1	SPI_2_SR[EOQF]
315	14EC	SPI2_2	SPI_2_SR[TFFF]
316	14F0	SPI2_3	SPI_2_SR[TCF]
317	14F4	SPI2_4	SPI_2_SR[RFDF]
318	14F8	SPI2_5	SPI_2_SR[CMD_TCF]
319	14FC	SPI2_6	SPI_2_SR[CMDFFF]
320	1500	SPI2_7	SPI_2_SR[SPEF]
322	1508	SPI3_0	SPI_3_SR[TFUF] | SPI_3_SR[RFOF]| SPI_3_SR[TFIWF]
323	150C	SPI3_1	SPI_3_SR[EOQF]
324	1510	SPI3_2	SPI_3_SR[TFFF]
325	1514	SPI3_3	SPI_3_SR[TCF]
326	1518	SPI3_4	SPI_3_SR[RFDF]
327	151C	SPI3_5	SPI_3_SR[SPITCF]
328	1520	SPI3_6	SPI_3_SR[DSITCF]
329	1524	SPI3_7	SPI_3_SR[SPEF] | SPI_3_SR[DPEF]
372	15D0	BCTU	LIST0_LAST | LIST1_LAST
373	15D4	BCTU	NDATA0 | NDATA1 | DATA_OVR0 | DATA_OVR1 | Trigger
376	15E0	LinFlex0_0	LinFlex_0_RXI
377	15E4	LinFlex0_1	LinFlex_0_TXI
378	15E8	LinFlex0_2	LinFlex_0_ERR
379	15EC	LinFlex1_0	LinFlex_1_RXI
380	15F0	LinFlex1_1	LinFlex_1_TXI
381	15F4	LinFlex1_2	LinFlex_1_ERR
382	15F8	LinFlex2_0	LinFlex_2_RXI
383	15FC	LinFlex2_1	LinFlex_2_TXI
384	1600	LinFlex2_2	LinFlex_2_ERR
385	1604	LinFlex3_0	LinFlex_3_RXI
386	1608	LinFlex3_1	LinFlex_3_TXI
387	160C	LinFlex3_2	LinFlex_3_ERR

Table continues on the next page...

Table 23-1. INTC Vector Table (continued)

Vector number	Vector offset	Source module	Source description
388	1610	LinFlex4_0	LinFlex_4_RXI
389	1614	LinFlex4_1	LinFlex_4_TXI
390	1618	LinFlex4_2	LinFlex_4_ERR
391	161C	LinFlex5_0	LinFlex_5_RXI
392	1620	LinFlex5_1	LinFlex_5_TXI
393	1624	LinFlex5_2	LinFlex_5_ERR
394	1628	LinFlex6_0	LinFlex_6_RXI
395	162C	LinFlex6_1	LinFlex_6_TXI
396	1630	LinFlex6_2	LinFlex_6_ERR
397	1634	LinFlex7_0	LinFlex_7_RXI
398	1638	LinFlex7_1	LinFlex_7_TXI
399	163C	LinFlex7_2	LinFlex_7_ERR
400	1640	LinFlex8_0	LinFlex_8_RXI
401	1644	LinFlex8_1	LinFlex_8_TXI
402	1648	LinFlex8_2	LinFlex_8_ERR
403	164C	LinFlex9_0	LinFlex_9_RXI
404	1650	LinFlex9_1	LinFlex_9_TXI
405	1654	LinFlex9_2	LinFlex_9_ERR
406	1658	LinFlex10_0	LinFlex_10_RXI
407	165C	LinFlex10_1	LinFlex_10_TXI
408	1660	LinFlex10_2	LinFlex_10_ERR
409	1664	LinFlex11_0	LinFlex_11_RXI
410	1668	LinFlex11_1	LinFlex_11_TXI
411	166C	LinFlex11_2	LinFlex_11_ERR
412	1670	LinFlex12_0	LinFlex_12_RXI
413	1674	LinFlex12_1	LinFlex_12_TXI
414	1678	LinFlex12_2	LinFlex_12_ERR
415	167C	LinFlex13_0	LinFlex_13_RXI
416	1680	LinFlex13_1	LinFlex_13_TXI
417	1684	LinFlex13_2	LinFlex_13_ERR
418	1688	LinFlex14_0	LinFlex_14_RXI
419	168C	LinFlex14_1	LinFlex_14_TXI
420	1690	LinFlex14_2	LinFlex_14_ERR
421	1694	LinFlex15_0	LinFlex_15_RXI
422	1698	LinFlex15_1	LinFlex_15_TXI
423	169C	LinFlex15_2	LinFlex_15_ERR
440	16E0	I2C_0_0	I2C0_SR[IBAL] | I2C0_SR[TCF] | I2C0_SR[IAAS]
442	16E8	I2C_1_0	I2C1_SR[IBAL] | I2C1_SR[TCF] | I2C1_SR[IAAS]
444	16F0	I2C_2_0	I2C2_SR[IBAL] | I2C2_SR[TCF] | I2C2_SR[IAAS]

Table continues on the next page...

Table 23-1. INTC Vector Table (continued)

Vector number	Vector offset	Source module	Source description
446	16F8	I2C_3_0	I2C3_SR[IBAL] | I2C3_SR[TCF] | I2C3_SR[IAAS]
453	1714	FlexRay_0_0	LRNEIF | DRNEIF
454	1718	FlexRay_0_1	LRCEIF | DRCEIF
455	171C	FlexRay_0_2	FAFAIF
456	1720	FlexRay_0_3	FAFBIF
457	1724	FlexRay_0_4	WUPIF
458	1728	FlexRay_0_5	PRIF
459	172C	FlexRay_0_6	CHIF
460	1730	FlexRay_0_7	TBIF
461	1734	FlexRay_0_8	RBIF
462	1738	FlexRay_0_9	MIF
480	1780	PLL	Loss-of-lock interrupt request for PLL
488	17A0	FCCU	Interrupt request (ALARM state)
489	17A4	FCCU	Interrupt request (miscellaneous conditions)
490	17A8	FCCU	EOUT interrupt
496	17C0	Hardware Security Module 0	HSM2HTIE[0]
497	17C4	Hardware Security Module 1	HSM2HTIE[1]
498	17C8	Hardware Security Module 2	HSM2HTIE[2]
499	17CC	Hardware Security Module 3	HSM2HTIE[3]
500	17D0	Hardware Security Module 4	HSM2HTIE[4]
501	17D4	Hardware Security Module 5	HSM2HTIE[5]
502	17D8	Hardware Security Module 6	HSM2HTIE[6]
503	17DC	Hardware Security Module 7	HSM2HTIE[7]
504	17E0	Hardware Security Module 8	HSM2HTIE[8]
505	17E4	Hardware Security Module 9	HSM2HTIE[9]
506	17E8	Hardware Security Module 10	HSM2HTIE[10]
507	17EC	Hardware Security Module 11	HSM2HTIE[11]
508	17F0	Hardware Security Module 12	HSM2HTIE[12]
509	17F4	Hardware Security Module 13	HSM2HTIE[13]
510	17F8	Hardware Security Module 14	HSM2HTIE[14]
511	17FC	Hardware Security Module 15	HSM2HTIE[15]
512	1800	Hardware Security Module 16	HSM2HTIE[16]
513	1804	Hardware Security Module 17	HSM2HTIE[17]
514	1808	Hardware Security Module 18	HSM2HTIE[18]
515	180C	Hardware Security Module 19	HSM2HTIE[19]
516	1810	Hardware Security Module 20	HSM2HTIE[20]
517	1814	Hardware Security Module 21	HSM2HTIE[21]
518	1818	Hardware Security Module 22	HSM2HTIE[22]
519	181C	Hardware Security Module 23	HSM2HTIE[23]

Table continues on the next page...

Table 23-1. INTC Vector Table (continued)

Vector number	Vector offset	Source module	Source description
520	1820	Hardware Security Module 24	HSM2HTIE[24]
521	1824	Hardware Security Module 25	HSM2HTIE[25]
522	1828	Hardware Security Module 26	HSM2HTIE[26]
523	182C	Hardware Security Module 27	HSM2HTIE[27]
524	1830	Hardware Security Module 28	HSM2HTIE[28]
525	1834	Hardware Security Module 29	HSM2HTIE[29]
526	1838	Hardware Security Module 30	HSM2HTIE[30]
527	183C	Hardware Security Module 31	HSM2HTIE[31]
548	1840	ADC_0_EOC	ADC_0_EOC
549	1890	ADC_0_ER	ADC_0_ER
550	1894	ADC_0_WD	ADC_0_WD
554	18A8	ADC_1_EOC	ADC_1_EOC
555	18AC	ADC_1_ER	ADC_1_ER
556	18B0	ADC_1_WD	ADC_1_WD
560	18C0	Analogue_Comparator_Module_0	CMP0
561	18C4	Analogue_Comparator_Module_1	CMP1
562	18C8	Analogue_Comparator_Module_2	CMP2
565	18D4	FlexCAN0_0	FlexCAN0_PN
566	18D8	FlexCAN0_1	FLEXCAN_0_ESR[ERR_INT] | FLEXCAN_0_FD_ERROR
567	18DC	FlexCAN0_2	FLEXCAN_0_ESR_BOFF | FLEXCAN_0_ESR_BOFF_DONE | FLEXCAN_0_Transmit_Warning | FLEXCAN_0_Receive_Warning
568	18E0	FlexCAN0_3	FlexCAN0_BUF_00_03
569	18E4	FlexCAN0_4	FlexCAN0_BUF_04_07
570	18E8	FlexCAN0_5	FlexCAN0_BUF_08_11
571	18EC	FlexCAN0_6	FlexCAN0_BUF_12_15
572	18F0	FlexCAN0_7	FlexCAN0_BUF_16_31
573	18F4	FlexCAN0_8	FlexCAN0_BUF_32_63
574	18F8	FlexCAN0_9	FlexCAN0_BUF_64_95
578	1908	FlexCAN1_0	FLEXCAN_1_ESR[ERR_INT] | FLEXCAN_1_FD_ERROR
579	190C	FlexCAN1_1	FLEXCAN_1_ESR_BOFF | FLEXCAN_1_ESR_BOFF_DONE | FLEXCAN_1_Transmit_Warning | FLEXCAN_1_Receive_Warning
580	1910	FlexCAN1_2	FlexCAN1_BUF_00_03
581	1914	FlexCAN1_3	FlexCAN1_BUF_04_07
582	1918	FlexCAN1_4	FlexCAN1_BUF_08_11
583	191C	FlexCAN1_5	FlexCAN1_BUF_12_15
584	1920	FlexCAN1_6	FlexCAN1_BUF_16_31

Table continues on the next page...

Table 23-1. INTC Vector Table (continued)

Vector number	Vector offset	Source module	Source description
585	1924	FlexCAN1_7	FlexCAN1_BUF_32_63
586	1928	FlexCAN1_8	FlexCAN1_BUF_64_95
590	1938	FlexCAN2_0	FLEXCAN_2_ESR[ERR_INT] | FLEXCAN_2_FD_ERROR
591	193C	FlexCAN2_1	FLEXCAN_2_ESR_BOFF | FLEXCAN_2_ESR_BOFF_DONE | FLEXCAN_2_Transmit_Warning | FLEXCAN_2_Receive_Warning
592	1940	FlexCAN2_2	FlexCAN2_BUF_00_03
593	1944	FlexCAN2_3	FlexCAN2_BUF_04_07
594	1948	FlexCAN2_4	FlexCAN2_BUF_08_11
595	194C	FlexCAN2_5	FlexCAN2_BUF_12_15
596	1950	FlexCAN2_6	FlexCAN2_BUF_16_31
597	1954	FlexCAN2_7	FlexCAN2_BUF_32_63
598	1958	FlexCAN2_8	FlexCAN2_BUF_64_95
602	1968	FlexCAN3_0	FLEXCAN_3_ESR[ERR_INT] | FLEXCAN_3_FD_ERROR
603	196C	FlexCAN3_1	FLEXCAN_3_ESR_BOFF | FLEXCAN_3_ESR_BOFF_DONE | FLEXCAN_3_Transmit_Warning | FLEXCAN_3_Receive_Warning
604	1970	FlexCAN3_2	FlexCAN3_BUF_00_03
605	1974	FlexCAN3_3	FlexCAN3_BUF_04_07
606	1978	FlexCAN3_4	FlexCAN3_BUF_08_11
607	197C	FlexCAN3_5	FlexCAN3_BUF_12_15
608	1980	FlexCAN3_6	FlexCAN3_BUF_16_31
609	1984	FlexCAN3_7	FlexCAN3_BUF_32_63
610	1988	FlexCAN3_8	FlexCAN3_BUF_64_95
614	1998	FlexCAN4_0	FLEXCAN_4_ESR[ERR_INT] | FLEXCAN_4_FD_ERROR
615	199C	FlexCAN4_1	FLEXCAN_4_ESR_BOFF | FLEXCAN_4_ESR_BOFF_DONE | FLEXCAN_4_Transmit_Warning | FLEXCAN_4_Receive_Warning
616	19A0	FlexCAN4_2	FlexCAN4_BUF_00_03
617	19A4	FlexCAN4_3	FlexCAN4_BUF_04_07
618	19A8	FlexCAN4_4	FlexCAN4_BUF_08_11
619	19AC	FlexCAN4_5	FlexCAN4_BUF_12_15
620	19B0	FlexCAN4_6	FlexCAN4_BUF_16_31
621	19B4	FlexCAN4_7	FlexCAN4_BUF_32_63
622	19B8	FlexCAN4_8	FlexCAN4_BUF_64_95
626	19C8	FlexCAN5_0	FLEXCAN_5_ESR[ERR_INT] | FLEXCAN_5_FD_ERROR

Table continues on the next page...

Table 23-1. INTC Vector Table (continued)

Vector number	Vector offset	Source module	Source description
627	19CC	FlexCAN5_1	FLEXCAN_5_ESR_BOFF | FLEXCAN_5_ESR_BOFF_DONE | FLEXCAN_5_Transmit_Warning | FLEXCAN_5_Receive_Warning
628	19D0	FlexCAN5_2	FlexCAN5_BUF_00_03
629	19D4	FlexCAN5_3	FlexCAN5_BUF_04_07
630	19D8	FlexCAN5_4	FlexCAN5BUF_08_11
631	19DC	FlexCAN5_5	FlexCAN5_BUF_12_15
632	19E0	FlexCAN5_6	FlexCAN5_BUF_16_31
633	19E4	FlexCAN5_7	FlexCAN5_BUF_32_63
634	19E8	FlexCAN5_8	FlexCAN5_BUF_64_95
638	19F8	FlexCAN6_0	FLEXCAN_6_ESR[ERR_INT] | FLEXCAN_6_FD_ERROR
639	19FC	FlexCAN6_1	FLEXCAN_6_ESR_BOFF | FLEXCAN_6_ESR_BOFF_DONE | FLEXCAN_6_Transmit_Warning | FLEXCAN_6_Receive_Warning
640	1A00	FlexCAN6_2	FlexCAN6_BUF_00_03
641	1A04	FlexCAN6_3	FlexCAN6_BUF_04_07
642	1A08	FlexCAN6_4	FlexCAN6_BUF_08_11
643	1A0C	FlexCAN6_5	FlexCAN6_BUF_12_15
644	1A10	FlexCAN6_6	FlexCAN6_BUF_16_31
645	1A14	FlexCAN6_7	FlexCAN6_BUF_32_63
646	1A18	FlexCAN6_8	FlexCAN6_BUF_64_95
650	1A28	FlexCAN7_0	FLEXCAN_7_ESR[ERR_INT] | FLEXCAN_7_FD_ERROR
651	1A2C	FlexCAN7_1	FLEXCAN_7_ESR_BOFF | FLEXCAN_7_ESR_BOFF_DONE | FLEXCAN_7_Transmit_Warning | FLEXCAN_7_Receive_Warning
652	1A30	FlexCAN7_2	FlexCAN7_BUF_00_03
653	1A34	FlexCAN7_3	FlexCAN7_BUF_04_07
654	1A38	FlexCAN7_4	FlexCAN7_BUF_08_11
655	1A3C	FlexCAN7_5	FlexCAN7_BUF_12_15
656	1A40	FlexCAN7_6	FlexCAN7_BUF_16_31
657	1A44	FlexCAN7_7	FlexCAN7_BUF_32_63
658	1A48	FlexCAN7_8	FlexCAN7_BUF_64_95
668	1A70	WKUP_IRQ_0	WKPU_IRQ_0
669	1A74	WKPU_IRQ_1	WKPU_IRQ_1
670	1A78	WKPU_IRQ_2	WKPU_IRQ_2
671	1A7C	WKPU_IRQ_3	WKPU_IRQ_3
690	1AC8	SAI0_TX_FIFO_IRQ	SAI0_TX_FIFO_IRQ
691	1ACC	SAI0_RX_FIFO_IRQ	SAI0_RX_FIFO_IRQ

Table continues on the next page...

Table 23-1. INTC Vector Table (continued)

Vector number	Vector offset	Source module	Source description
692	1AD0	SAI1_TX_FIFO_IRQ	SAI1_TX_FIFO_IRQ
693	1AD4	SAI1_RX_FIFO_IRQ	SAI1_RX_FIFO_IRQ
694	1AD8	SAI2_TX_FIFO_IRQ	SAI2_TX_FIFO_IRQ
695	1ADC	SAI2_RX_FIFO_IRQ	SAI2_RX_FIFO_IRQ
701	1AF4	JDC	JDC
702	1AFC	MEMU_1	Correctable errors in LPU mode
706	1B08	eMIOS 0 channel 0,1	EMIOS_GFR[F0,F1]
707	1B0C	eMIOS 0 channel 2,3	EMIOS_GFR[F2,F3]
708	1B10	eMIOS 0 channel 4,5	EMIOS_GFR[F4,F5]
709	1B14	eMIOS 0 channel 6,7	EMIOS_GFR[F6,F7]
710	1B18	eMIOS 0 channel 8,9	EMIOS_GFR[F8,F9]
711	1B1C	eMIOS 0 channel 10,11	EMIOS_GFR[F10,F11]
712	1B20	eMIOS 0 channel 12,13	EMIOS_GFR[F12,F13]
713	1B24	eMIOS 0 channel 14,15	EMIOS_GFR[F14,F15]
714	1B28	eMIOS 0 channel 16,17	EMIOS_GFR[F16,F17]
715	1B2C	eMIOS 0 channel 18,19	EMIOS_GFR[F18,F19]
716	1B30	eMIOS 0 channel 20,21	EMIOS_GFR[F20,F21]
717	1B34	eMIOS 0 channel 22,23	EMIOS_GFR[F22,F23]
718	1B38	eMIOS 0 channel 24,25	EMIOS_GFR[F24,F25]
719	1B3C	eMIOS 0 channel 26,27	EMIOS_GFR[F26,F27]
720	1B40	eMIOS 0 channel 28,29	EMIOS_GFR[F28,F29]
721	1B44	eMIOS 0 channel 30,31	EMIOS_GFR[F30,F31]
722	1B48	eMIOS 1 channel 0,1	EMIOS_GFR[F32,F33]
723	1B4C	eMIOS 1 channel 2, 3	EMIOS_GFR[F34,F35]
724	1B50	eMIOS 1 channel 4, 5	EMIOS_GFR[F36,F37]
725	1B54	eMIOS 1 channel 6, 7	EMIOS_GFR[F38,F39]
726	1B58	eMIOS 1 channel 8, 9	EMIOS_GFR[F40,F41]
727	1B5C	eMIOS 1 channel 10, 11	EMIOS_GFR[F42,F43]
728	1B60	eMIOS 1 channel 12, 13	EMIOS_GFR[F44,F45]
729	1B64	eMIOS 1 channel 14, 15	EMIOS_GFR[F46,F47]
730	1B68	eMIOS 1 channel 16, 17	EMIOS_GFR[F48,F49]
731	1B6C	eMIOS 1 channel 18, 19	EMIOS_GFR[F50,F51]
732	1B70	eMIOS 1 channel 20, 21	EMIOS_GFR[F52,F53]
733	1B74	eMIOS 1 channel 22, 23	EMIOS_GFR[F54,F55]
734	1B78	eMIOS 1 channel 24, 25	EMIOS_GFR[F56,F57]
735	1B7C	eMIOS 1 channel 26, 27	EMIOS_GFR[F58,F59]
736	1B80	eMIOS 1 channel 28, 29	EMIOS_GFR[F60,F61]
737	1B84	eMIOS 1 channel 30, 31	EMIOS_GFR[F62,F63]
754	1BE0	HFIF_0	ACK Interrupt

Table continues on the next page...

Table 23-1. INTC Vector Table (continued)

Vector number	Vector offset	Source module	Source description
755	1BE4	HFIF_0	INIT Interrupt
756	1BE8	HFIF_0	EOF Interrupt
757	1BEC	HFIF_1	ACK Interrupt
758	1BF0	HFIF_1	INIT Interrupt
759	1BF4	HFIF_1	EOF Interrupt
760	1BF8	HFIF_2	ACK Interrupt
761	1BFC	HFIF_2	INIT Interrupt
762	1C00	HFIF_2	EOF Interrupt
763	1C04	HFIF_3	ACK Interrupt
764	1C08	HFIF_3	INIT Interrupt
765	1C0C	HFIF_3	EOF Interrupt

NOTE

- Transfer error would not be generated if 477 and 484 slot is accessed.
- Interrupts are not serviced during self-test.

23.1.3 INTC implemented registers

The following table shows the registers implemented on the INTC on this chip. All offsets not specifically listed are reserved. In this chapter, z4and z2 cores are represented as processor 0, processor 1respectively.

Table 23-2. INTC implemented registers

Address offset	Register
000h	INTC Block Configuration Register (INTC_BCR)
010h	INTC Current Priority Register for Processor 0 (INTC_CPR0)
014h	INTC Current Priority Register for Processor 1 (INTC_CPR1)
018h	Reserved
020h	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)
024h	INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)
028h	Reserved
030h	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)
034h	INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)
038h	Reserved

Table continues on the next page...

Table 23-2. INTC implemented registers (continued)

Address offset	Register
040h–057h	INTC Software Set/Clear Interrupt Register 0 (INTC_SSCIR0) – INTC Software Set/Clear Interrupt Register 23 (INTC_SSCIR23)
060h-5FAh	INTC Priority Select Register 24 (INTC_PSR24) – INTC Priority Select Register 765 (INTC_PSR765)

23.1.4 Interrupts handling during mode transitions

1. All interrupts which are masked in INTC should also be marked as disabled in the respective modules.
2. All the modules, which do not support STOP-ACK handshake in hardware preferably should have the interrupt routines to poll for MC_ME_GS[S_MTRANS] to be 0 or for such modules the interrupts should be disabled before initiating mode transition request.
3. Modules which support STOP-ACK handshake (ADC, FlexCAN, DSPI, i2c, LINFlexD, SAI, RTC, BCTU, ENET and PIT-RTI) should have the interrupt service routine such that the interrupt flag in the module (i.e Program SSCIRn[CLR] to 1) is cleared and then before accessing any other module it waits for MC_ME_GS[S_MTRANS] to go to 0.

23.2 Introduction

The INTC:

- Provides priority-based preemptive scheduling of interrupt requests
- Schedules interrupt requests (IRQs) from software and internal peripherals to one or more processors (PRCs)
- Provides interrupt prioritization and preemption, interrupt masking, interrupt priority elevation, and protocol support

This scheduling scheme is suitable for statically scheduled hard real-time systems.

The INTC is targeted to work with a Power Architecture processor and is capable of processing high-demand interrupt sources where the Interrupt Service Routines (ISRs) nest to multiple levels, but it also can be used with other processors and applications.

For high-priority interrupt requests in these target applications, it is necessary to minimize the interval between the assertion of the interrupt request from the peripheral and the point at which the processor is performing useful work to service the interrupt request. The INTC supports this goal by providing a unique vector for each interrupt request source. It also provides 16 priorities so that lower priority ISRs do not delay the execution of higher priority ISRs. Since each individual application will have different priorities for each source of interrupt request, the priority of each interrupt request is configurable.

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

Multiple processors can assert interrupt requests to each other through software-settable interrupt requests. These same software-settable interrupt requests can also be used to separate the work involved in servicing an interrupt request into two parts, a high-priority portion and a low-priority portion. The high-priority portion is initiated by a peripheral interrupt request, but then the ISR can assert a software-settable interrupt request to finish the servicing in a lower priority ISR. Therefore these software-settable interrupt requests can be used instead of having the peripheral ISR schedule a task through the RTOS.

23.3 Block diagram

The following figure shows the block diagram of an INTC with four processors (PRC0...PRC3). The actual number of processors is described in the chip-specific INTC information. The structure of the INTC remains the same for each implemented processor.

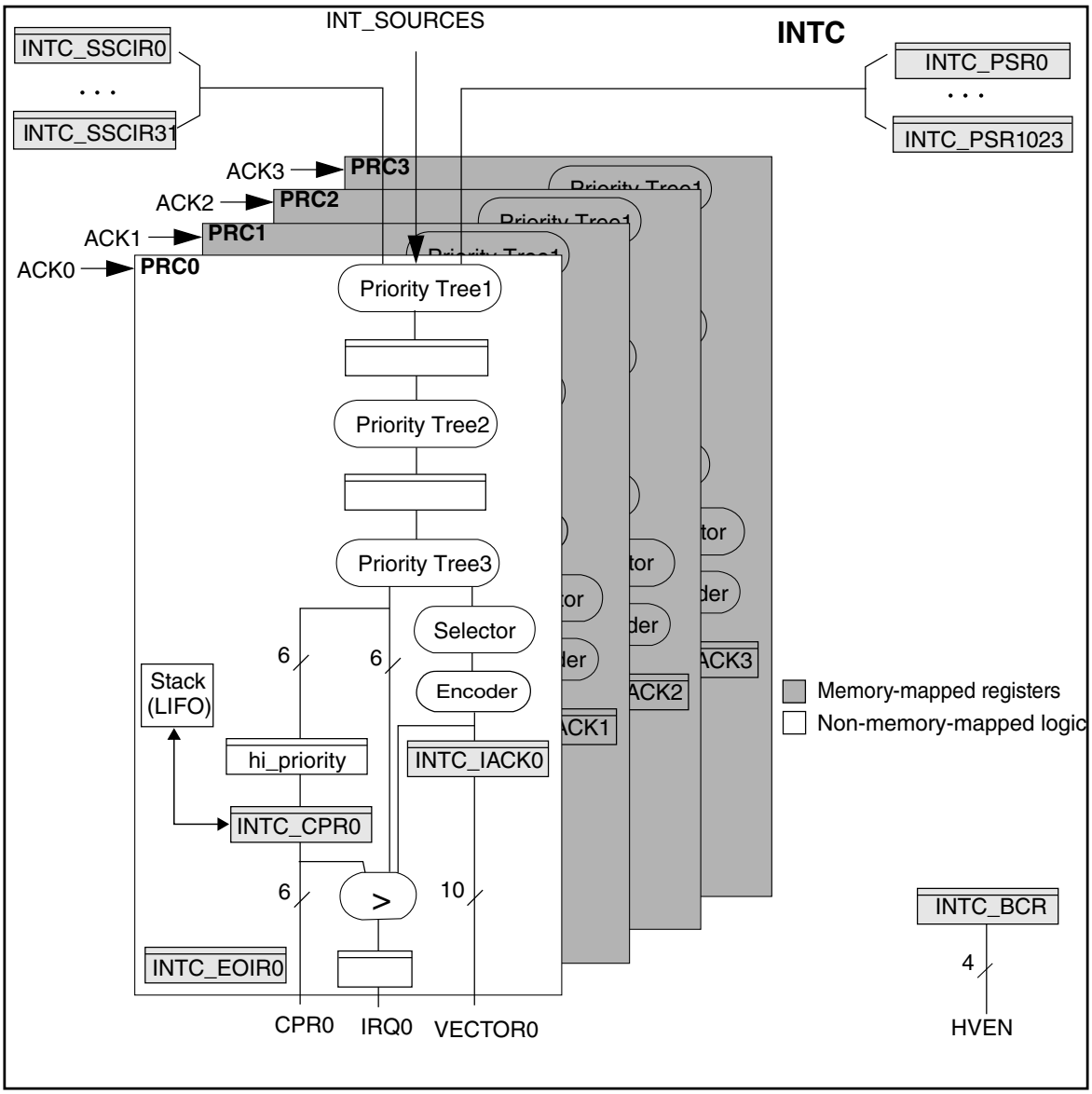


Figure 23-1. Block diagram for an INTC with four processors

23.4 Features

- Each peripheral interrupt source is software-steerable to any combination of interrupt request outputs to the following:
  - Processor 0
  - Processor 1
- 24 software-settable interrupt request sources
- 10-bit vector

    - Unique vector for each interrupt request source
    - Hardware connection to processor or read from register
- Each interrupt source can be programmed to one of 16 priorities
- Each interrupt source can be triggered by software
- Preemption
    - Preemptive prioritized interrupt requests to processor
    - ISR with higher priority preempts ISRs or tasks with lower priorities
    - Automatic pushing or popping of preempted priority to or from a LIFO
    - Ability to modify the ISR or task priority; modifying the priority can be used to implement the Priority Ceiling Protocol for accessing shared resources
    - 3 INTC clock cycles from interrupt request into interrupt request to CPU
- Low latency
    - Two INTC clock cycles from receipt of interrupt request from peripheral to interrupt request to processor; three clock cycles from receipt of software request

23.5 Modes of operation

The interrupt controller has two handshaking modes with the processor: software vector mode and hardware vector mode. The state of the hardware vector enable bit, INTC_BCR[HVEN], determines which mode is used. In Power Architecture devices, software vector mode uses the autovector mode of the Processor for a single entry point for the ISR (16 bytes of vector space available), whereas the hardware vector mode uses the non-autovector mode where the vector offset from the INTC is provided to the Processor (4 bytes for each IRQ).

In debug mode the interrupt controller operation is identical to its normal operation of software vector mode or hardware vector mode.

23.5.1 Software vector mode

In software vector mode, software (that is, the interrupt exception handler) must read a register in the INTC to obtain the vector associated with the interrupt request to the processor. The INTC will use software vector mode for a given processor when its

associated HVEN_PRCn bit in the INTC_BCR is negated. The hardware vector enable signal to any processor is driven as negated when its associated HVEN_PRC*n* bit is negated. The vector is read from the INTC_IACKRn registers. Reading the INTC_IACKR*n* negates the interrupt request to the associated processor. Even if a higher priority interrupt request has arrived while waiting for this interrupt acknowledge, the interrupt request to the processor will negate for at least one clock. The reading also pushes the PRI value in the INTC_CPRn onto the associated LIFO, and updates PRI in the associated INTC_CPR_PRC*n* with the new priority.

23.5.2 Hardware vector mode

In hardware vector mode, the hardware causes the first instruction that will be executed (when handling the interrupt request to the processor) to be an instruction that is specific to that vector. Therefore the interrupt exception handler is specific to a peripheral or software-settable interrupt request, rather than being common to all of them. The INTC will use hardware vector mode for a given processor when its associated HVENn bit in the INTC_BCR is asserted. The hardware vector enable signal to the associated processor is driven as asserted.

When the interrupt request to the associated processor asserts, the interrupt vector signal is updated. The value of that interrupt vector is the unique vector associated with the preempting peripheral or software-settable interrupt request. The vector value matches the value of the INTVEC field in the INTC_IACKRn, depending on which processor was assigned to handle a given interrupt source.

The assertion of the interrupt acknowledge signal for a given processor pushes the associated PRI value in the associated INTC_CPR*n* register onto the associated LIFO and updates the associated PRI in the associated INTC_CPR*n* register with the new priority. This pushing of the PRI value onto the associated LIFO and updating PRI in the associated INTC_CPR*n* does not occur when the associated interrupt acknowledge signal asserts and the INTC_SSCIRs is written at a time such that the PRI value in the associated INTC_CPR*n* register would need to be pushed and the previously last pushed PRI value would need to be popped simultaneously. In this case, PRI in the associated INTC_CPR*n* is updated with the new priority, and the associated LIFO is neither pushed or popped.

23.6 Memory map and register definition

A transfer error will be asserted if an access is attempted outside of the memory map or for any unimplemented registers. For example, if the design has only 512 interrupt sources and a source > 512 is accessed, a transfer error is asserted.

With the exception of the INTC_SSCIR *n* and INTC_PSR *n* registers, all of the registers are 32-bit. Any combination of accessing the four bytes of a register with a single access is supported, provided that the access does not cross a register boundary. These supported accesses include types and sizes of 8 bits, aligned 16 bits, misaligned 16 bits to the middle two bytes, and aligned 32 bits.

Although INTC_SSCIR*n* are 8 bits wide and INTC_PSR*n* are 16 bits wide, they can be accessed with a single 16-bit or 32-bit access, provided that the access does not cross a 32-bit boundary.

Some registers have specific exceptions to these rules, as outlined in their definitions.

When writing to INTC_PSRs or INTC_SSCIRs, the following restrictions (illustrated in Figure 23-2) apply:

- For PSRs, write accesses must not span 16-bit boundaries where one register is implemented and one is unimplemented (unimplemented interrupt source).
- For SSCIRs, write accesses must not span 8-bit boundaries where one register is implemented and one is unimplemented (unimplemented interrupt source).

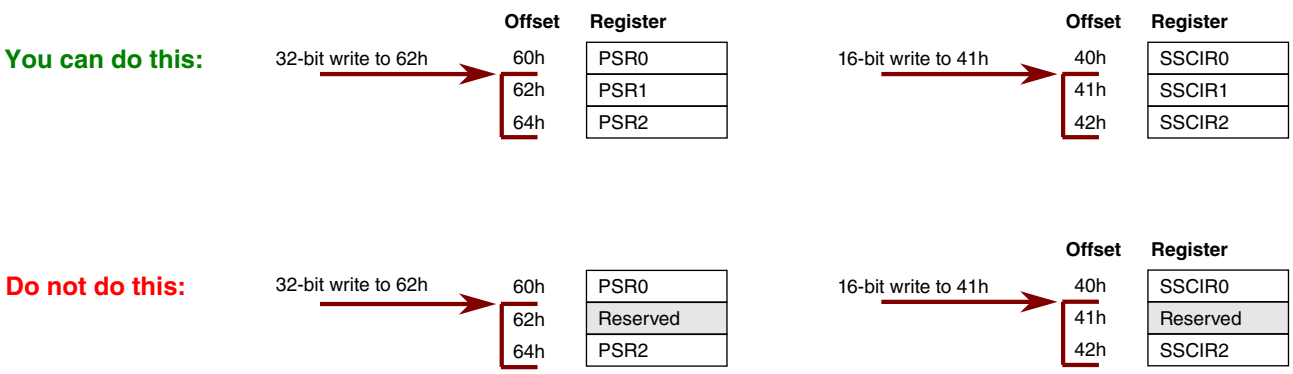


Figure 23-2. INTC_PSR*n* and INTC_SSCIR*n* write restrictions

In software vector mode, the effects of a read of the INTC Interrupt Acknowledge register (INTC_IACKR *n*) are the same regardless of the size of the read. In either software or hardware vector mode, the size of a write to the INTC end-of-interrupt register (INTC_EOIR *n*) does not affect the operation of the write.

INTC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	INTC Block Configuration Register (INTC_BCR)	32	R/W	0000_0000h	23.6.1/534

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	INTC Master Protection Register (INTC_MPROT)	32	R/W	0000_0000h	23.6.2/535
10	INTC Current Priority Register for Processor 0 (INTC_CPR0)	32	R/W	0000_000Fh	23.6.3/536
14	INTC Current Priority Register for Processor 1 (INTC_CPR1)	32	R/W	0000_000Fh	23.6.4/537
20	INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)	32	R/W	0000_0000h	23.6.5/538
24	INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)	32	R/W	0000_0000h	23.6.6/539
30	INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)	32	W	0000_0000h	23.6.7/540
34	INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)	32	W	0000_0000h	23.6.8/541
40	INTC Software Set/Clear Interrupt Register (INTC_SSCIR0)	8	R/W	00h	23.6.9/541
41	INTC Software Set/Clear Interrupt Register (INTC_SSCIR1)	8	R/W	00h	23.6.9/541
42	INTC Software Set/Clear Interrupt Register (INTC_SSCIR2)	8	R/W	00h	23.6.9/541
43	INTC Software Set/Clear Interrupt Register (INTC_SSCIR3)	8	R/W	00h	23.6.9/541
44	INTC Software Set/Clear Interrupt Register (INTC_SSCIR4)	8	R/W	00h	23.6.9/541
45	INTC Software Set/Clear Interrupt Register (INTC_SSCIR5)	8	R/W	00h	23.6.9/541
46	INTC Software Set/Clear Interrupt Register (INTC_SSCIR6)	8	R/W	00h	23.6.9/541
47	INTC Software Set/Clear Interrupt Register (INTC_SSCIR7)	8	R/W	00h	23.6.9/541
48	INTC Software Set/Clear Interrupt Register (INTC_SSCIR8)	8	R/W	00h	23.6.9/541
49	INTC Software Set/Clear Interrupt Register (INTC_SSCIR9)	8	R/W	00h	23.6.9/541
4A	INTC Software Set/Clear Interrupt Register (INTC_SSCIR10)	8	R/W	00h	23.6.9/541
4B	INTC Software Set/Clear Interrupt Register (INTC_SSCIR11)	8	R/W	00h	23.6.9/541
4C	INTC Software Set/Clear Interrupt Register (INTC_SSCIR12)	8	R/W	00h	23.6.9/541
4D	INTC Software Set/Clear Interrupt Register (INTC_SSCIR13)	8	R/W	00h	23.6.9/541
4E	INTC Software Set/Clear Interrupt Register (INTC_SSCIR14)	8	R/W	00h	23.6.9/541
4F	INTC Software Set/Clear Interrupt Register (INTC_SSCIR15)	8	R/W	00h	23.6.9/541
50	INTC Software Set/Clear Interrupt Register (INTC_SSCIR16)	8	R/W	00h	23.6.9/541
51	INTC Software Set/Clear Interrupt Register (INTC_SSCIR17)	8	R/W	00h	23.6.9/541
52	INTC Software Set/Clear Interrupt Register (INTC_SSCIR18)	8	R/W	00h	23.6.9/541
53	INTC Software Set/Clear Interrupt Register (INTC_SSCIR19)	8	R/W	00h	23.6.9/541
54	INTC Software Set/Clear Interrupt Register (INTC_SSCIR20)	8	R/W	00h	23.6.9/541

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
55	INTC Software Set/Clear Interrupt Register (INTC_SSCIR21)	8	R/W	00h	23.6.9/541
56	INTC Software Set/Clear Interrupt Register (INTC_SSCIR22)	8	R/W	00h	23.6.9/541
57	INTC Software Set/Clear Interrupt Register (INTC_SSCIR23)	8	R/W	00h	23.6.9/541
60	INTC Priority Select Register (INTC_PSR0)	16	R/W	8000h	23.6.10/543
62	INTC Priority Select Register (INTC_PSR1)	16	R/W	8000h	23.6.10/543
64	INTC Priority Select Register (INTC_PSR2)	16	R/W	8000h	23.6.10/543
66	INTC Priority Select Register (INTC_PSR3)	16	R/W	8000h	23.6.10/543
68	INTC Priority Select Register (INTC_PSR4)	16	R/W	8000h	23.6.10/543
6A	INTC Priority Select Register (INTC_PSR5)	16	R/W	8000h	23.6.10/543
6C	INTC Priority Select Register (INTC_PSR6)	16	R/W	8000h	23.6.10/543
6E	INTC Priority Select Register (INTC_PSR7)	16	R/W	8000h	23.6.10/543
70	INTC Priority Select Register (INTC_PSR8)	16	R/W	8000h	23.6.10/543
72	INTC Priority Select Register (INTC_PSR9)	16	R/W	8000h	23.6.10/543
74	INTC Priority Select Register (INTC_PSR10)	16	R/W	8000h	23.6.10/543
76	INTC Priority Select Register (INTC_PSR11)	16	R/W	8000h	23.6.10/543
78	INTC Priority Select Register (INTC_PSR12)	16	R/W	8000h	23.6.10/543
7A	INTC Priority Select Register (INTC_PSR13)	16	R/W	8000h	23.6.10/543
7C	INTC Priority Select Register (INTC_PSR14)	16	R/W	8000h	23.6.10/543
7E	INTC Priority Select Register (INTC_PSR15)	16	R/W	8000h	23.6.10/543
80	INTC Priority Select Register (INTC_PSR16)	16	R/W	8000h	23.6.10/543
82	INTC Priority Select Register (INTC_PSR17)	16	R/W	8000h	23.6.10/543
84	INTC Priority Select Register (INTC_PSR18)	16	R/W	8000h	23.6.10/543
86	INTC Priority Select Register (INTC_PSR19)	16	R/W	8000h	23.6.10/543
88	INTC Priority Select Register (INTC_PSR20)	16	R/W	8000h	23.6.10/543
8A	INTC Priority Select Register (INTC_PSR21)	16	R/W	8000h	23.6.10/543
8C	INTC Priority Select Register (INTC_PSR22)	16	R/W	8000h	23.6.10/543
8E	INTC Priority Select Register (INTC_PSR23)	16	R/W	8000h	23.6.10/543
90	INTC Priority Select Register (INTC_PSR24)	16	R/W	8000h	23.6.10/543
92	INTC Priority Select Register (INTC_PSR25)	16	R/W	8000h	23.6.10/543
94	INTC Priority Select Register (INTC_PSR26)	16	R/W	8000h	23.6.10/543
96	INTC Priority Select Register (INTC_PSR27)	16	R/W	8000h	23.6.10/543
98	INTC Priority Select Register (INTC_PSR28)	16	R/W	8000h	23.6.10/543
9A	INTC Priority Select Register (INTC_PSR29)	16	R/W	8000h	23.6.10/543
9C	INTC Priority Select Register (INTC_PSR30)	16	R/W	8000h	23.6.10/543
9E	INTC Priority Select Register (INTC_PSR31)	16	R/W	8000h	23.6.10/543
A0	INTC Priority Select Register (INTC_PSR32)	16	R/W	8000h	23.6.10/543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A2	INTC Priority Select Register (INTC_PSR33)	16	R/W	8000h	23.6.10/543
A4	INTC Priority Select Register (INTC_PSR34)	16	R/W	8000h	23.6.10/543
A6	INTC Priority Select Register (INTC_PSR35)	16	R/W	8000h	23.6.10/543
A8	INTC Priority Select Register (INTC_PSR36)	16	R/W	8000h	23.6.10/543
AA	INTC Priority Select Register (INTC_PSR37)	16	R/W	8000h	23.6.10/543
AC	INTC Priority Select Register (INTC_PSR38)	16	R/W	8000h	23.6.10/543
AE	INTC Priority Select Register (INTC_PSR39)	16	R/W	8000h	23.6.10/543
B0	INTC Priority Select Register (INTC_PSR40)	16	R/W	8000h	23.6.10/543
B2	INTC Priority Select Register (INTC_PSR41)	16	R/W	8000h	23.6.10/543
B4	INTC Priority Select Register (INTC_PSR42)	16	R/W	8000h	23.6.10/543
B6	INTC Priority Select Register (INTC_PSR43)	16	R/W	8000h	23.6.10/543
B8	INTC Priority Select Register (INTC_PSR44)	16	R/W	8000h	23.6.10/543
BA	INTC Priority Select Register (INTC_PSR45)	16	R/W	8000h	23.6.10/543
BC	INTC Priority Select Register (INTC_PSR46)	16	R/W	8000h	23.6.10/543
BE	INTC Priority Select Register (INTC_PSR47)	16	R/W	8000h	23.6.10/543
C0	INTC Priority Select Register (INTC_PSR48)	16	R/W	8000h	23.6.10/543
C2	INTC Priority Select Register (INTC_PSR49)	16	R/W	8000h	23.6.10/543
C4	INTC Priority Select Register (INTC_PSR50)	16	R/W	8000h	23.6.10/543
C6	INTC Priority Select Register (INTC_PSR51)	16	R/W	8000h	23.6.10/543
C8	INTC Priority Select Register (INTC_PSR52)	16	R/W	8000h	23.6.10/543
CA	INTC Priority Select Register (INTC_PSR53)	16	R/W	8000h	23.6.10/543
CC	INTC Priority Select Register (INTC_PSR54)	16	R/W	8000h	23.6.10/543
CE	INTC Priority Select Register (INTC_PSR55)	16	R/W	8000h	23.6.10/543
D0	INTC Priority Select Register (INTC_PSR56)	16	R/W	8000h	23.6.10/543
D2	INTC Priority Select Register (INTC_PSR57)	16	R/W	8000h	23.6.10/543
D4	INTC Priority Select Register (INTC_PSR58)	16	R/W	8000h	23.6.10/543
D6	INTC Priority Select Register (INTC_PSR59)	16	R/W	8000h	23.6.10/543
D8	INTC Priority Select Register (INTC_PSR60)	16	R/W	8000h	23.6.10/543
DA	INTC Priority Select Register (INTC_PSR61)	16	R/W	8000h	23.6.10/543
DC	INTC Priority Select Register (INTC_PSR62)	16	R/W	8000h	23.6.10/543
DE	INTC Priority Select Register (INTC_PSR63)	16	R/W	8000h	23.6.10/543
E0	INTC Priority Select Register (INTC_PSR64)	16	R/W	8000h	23.6.10/543
E2	INTC Priority Select Register (INTC_PSR65)	16	R/W	8000h	23.6.10/543
E4	INTC Priority Select Register (INTC_PSR66)	16	R/W	8000h	23.6.10/543
E6	INTC Priority Select Register (INTC_PSR67)	16	R/W	8000h	23.6.10/543
E8	INTC Priority Select Register (INTC_PSR68)	16	R/W	8000h	23.6.10/543
EA	INTC Priority Select Register (INTC_PSR69)	16	R/W	8000h	23.6.10/543
EC	INTC Priority Select Register (INTC_PSR70)	16	R/W	8000h	23.6.10/543
EE	INTC Priority Select Register (INTC_PSR71)	16	R/W	8000h	23.6.10/543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F0	INTC Priority Select Register (INTC_PSR72)	16	R/W	8000h	23.6.10/543
F2	INTC Priority Select Register (INTC_PSR73)	16	R/W	8000h	23.6.10/543
F4	INTC Priority Select Register (INTC_PSR74)	16	R/W	8000h	23.6.10/543
F6	INTC Priority Select Register (INTC_PSR75)	16	R/W	8000h	23.6.10/543
F8	INTC Priority Select Register (INTC_PSR76)	16	R/W	8000h	23.6.10/543
FA	INTC Priority Select Register (INTC_PSR77)	16	R/W	8000h	23.6.10/543
FC	INTC Priority Select Register (INTC_PSR78)	16	R/W	8000h	23.6.10/543
FE	INTC Priority Select Register (INTC_PSR79)	16	R/W	8000h	23.6.10/543
100	INTC Priority Select Register (INTC_PSR80)	16	R/W	8000h	23.6.10/543
102	INTC Priority Select Register (INTC_PSR81)	16	R/W	8000h	23.6.10/543
104	INTC Priority Select Register (INTC_PSR82)	16	R/W	8000h	23.6.10/543
106	INTC Priority Select Register (INTC_PSR83)	16	R/W	8000h	23.6.10/543
108	INTC Priority Select Register (INTC_PSR84)	16	R/W	8000h	23.6.10/543
10A	INTC Priority Select Register (INTC_PSR85)	16	R/W	8000h	23.6.10/543
10C	INTC Priority Select Register (INTC_PSR86)	16	R/W	8000h	23.6.10/543
10E	INTC Priority Select Register (INTC_PSR87)	16	R/W	8000h	23.6.10/543
110	INTC Priority Select Register (INTC_PSR88)	16	R/W	8000h	23.6.10/543
112	INTC Priority Select Register (INTC_PSR89)	16	R/W	8000h	23.6.10/543
114	INTC Priority Select Register (INTC_PSR90)	16	R/W	8000h	23.6.10/543
116	INTC Priority Select Register (INTC_PSR91)	16	R/W	8000h	23.6.10/543
118	INTC Priority Select Register (INTC_PSR92)	16	R/W	8000h	23.6.10/543
11A	INTC Priority Select Register (INTC_PSR93)	16	R/W	8000h	23.6.10/543
11C	INTC Priority Select Register (INTC_PSR94)	16	R/W	8000h	23.6.10/543
11E	INTC Priority Select Register (INTC_PSR95)	16	R/W	8000h	23.6.10/543
120	INTC Priority Select Register (INTC_PSR96)	16	R/W	8000h	23.6.10/543
122	INTC Priority Select Register (INTC_PSR97)	16	R/W	8000h	23.6.10/543
124	INTC Priority Select Register (INTC_PSR98)	16	R/W	8000h	23.6.10/543
126	INTC Priority Select Register (INTC_PSR99)	16	R/W	8000h	23.6.10/543
128	INTC Priority Select Register (INTC_PSR100)	16	R/W	8000h	23.6.10/543
12A	INTC Priority Select Register (INTC_PSR101)	16	R/W	8000h	23.6.10/543
12C	INTC Priority Select Register (INTC_PSR102)	16	R/W	8000h	23.6.10/543
12E	INTC Priority Select Register (INTC_PSR103)	16	R/W	8000h	23.6.10/543
130	INTC Priority Select Register (INTC_PSR104)	16	R/W	8000h	23.6.10/543
132	INTC Priority Select Register (INTC_PSR105)	16	R/W	8000h	23.6.10/543
134	INTC Priority Select Register (INTC_PSR106)	16	R/W	8000h	23.6.10/543
136	INTC Priority Select Register (INTC_PSR107)	16	R/W	8000h	23.6.10/543
138	INTC Priority Select Register (INTC_PSR108)	16	R/W	8000h	23.6.10/543
13A	INTC Priority Select Register (INTC_PSR109)	16	R/W	8000h	23.6.10/543
13C	INTC Priority Select Register (INTC_PSR110)	16	R/W	8000h	23.6.10/543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13E	INTC Priority Select Register (INTC_PSR111)	16	R/W	8000h	23.6.10/543
140	INTC Priority Select Register (INTC_PSR112)	16	R/W	8000h	23.6.10/543
142	INTC Priority Select Register (INTC_PSR113)	16	R/W	8000h	23.6.10/543
144	INTC Priority Select Register (INTC_PSR114)	16	R/W	8000h	23.6.10/543
146	INTC Priority Select Register (INTC_PSR115)	16	R/W	8000h	23.6.10/543
148	INTC Priority Select Register (INTC_PSR116)	16	R/W	8000h	23.6.10/543
14A	INTC Priority Select Register (INTC_PSR117)	16	R/W	8000h	23.6.10/543
14C	INTC Priority Select Register (INTC_PSR118)	16	R/W	8000h	23.6.10/543
14E	INTC Priority Select Register (INTC_PSR119)	16	R/W	8000h	23.6.10/543
150	INTC Priority Select Register (INTC_PSR120)	16	R/W	8000h	23.6.10/543
152	INTC Priority Select Register (INTC_PSR121)	16	R/W	8000h	23.6.10/543
154	INTC Priority Select Register (INTC_PSR122)	16	R/W	8000h	23.6.10/543
156	INTC Priority Select Register (INTC_PSR123)	16	R/W	8000h	23.6.10/543
158	INTC Priority Select Register (INTC_PSR124)	16	R/W	8000h	23.6.10/543
15A	INTC Priority Select Register (INTC_PSR125)	16	R/W	8000h	23.6.10/543
15C	INTC Priority Select Register (INTC_PSR126)	16	R/W	8000h	23.6.10/543
15E	INTC Priority Select Register (INTC_PSR127)	16	R/W	8000h	23.6.10/543
160	INTC Priority Select Register (INTC_PSR128)	16	R/W	8000h	23.6.10/543
162	INTC Priority Select Register (INTC_PSR129)	16	R/W	8000h	23.6.10/543
164	INTC Priority Select Register (INTC_PSR130)	16	R/W	8000h	23.6.10/543
166	INTC Priority Select Register (INTC_PSR131)	16	R/W	8000h	23.6.10/543
168	INTC Priority Select Register (INTC_PSR132)	16	R/W	8000h	23.6.10/543
16A	INTC Priority Select Register (INTC_PSR133)	16	R/W	8000h	23.6.10/543
16C	INTC Priority Select Register (INTC_PSR134)	16	R/W	8000h	23.6.10/543
16E	INTC Priority Select Register (INTC_PSR135)	16	R/W	8000h	23.6.10/543
170	INTC Priority Select Register (INTC_PSR136)	16	R/W	8000h	23.6.10/543
172	INTC Priority Select Register (INTC_PSR137)	16	R/W	8000h	23.6.10/543
174	INTC Priority Select Register (INTC_PSR138)	16	R/W	8000h	23.6.10/543
176	INTC Priority Select Register (INTC_PSR139)	16	R/W	8000h	23.6.10/543
178	INTC Priority Select Register (INTC_PSR140)	16	R/W	8000h	23.6.10/543
17A	INTC Priority Select Register (INTC_PSR141)	16	R/W	8000h	23.6.10/543
17C	INTC Priority Select Register (INTC_PSR142)	16	R/W	8000h	23.6.10/543
17E	INTC Priority Select Register (INTC_PSR143)	16	R/W	8000h	23.6.10/543
180	INTC Priority Select Register (INTC_PSR144)	16	R/W	8000h	23.6.10/543
182	INTC Priority Select Register (INTC_PSR145)	16	R/W	8000h	23.6.10/543
184	INTC Priority Select Register (INTC_PSR146)	16	R/W	8000h	23.6.10/543
186	INTC Priority Select Register (INTC_PSR147)	16	R/W	8000h	23.6.10/543
188	INTC Priority Select Register (INTC_PSR148)	16	R/W	8000h	23.6.10/543
18A	INTC Priority Select Register (INTC_PSR149)	16	R/W	8000h	23.6.10/543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
18C	INTC Priority Select Register (INTC_PSR150)	16	R/W	8000h	23.6.10/543
18E	INTC Priority Select Register (INTC_PSR151)	16	R/W	8000h	23.6.10/543
190	INTC Priority Select Register (INTC_PSR152)	16	R/W	8000h	23.6.10/543
192	INTC Priority Select Register (INTC_PSR153)	16	R/W	8000h	23.6.10/543
194	INTC Priority Select Register (INTC_PSR154)	16	R/W	8000h	23.6.10/543
196	INTC Priority Select Register (INTC_PSR155)	16	R/W	8000h	23.6.10/543
198	INTC Priority Select Register (INTC_PSR156)	16	R/W	8000h	23.6.10/543
19A	INTC Priority Select Register (INTC_PSR157)	16	R/W	8000h	23.6.10/543
19C	INTC Priority Select Register (INTC_PSR158)	16	R/W	8000h	23.6.10/543
19E	INTC Priority Select Register (INTC_PSR159)	16	R/W	8000h	23.6.10/543
1A0	INTC Priority Select Register (INTC_PSR160)	16	R/W	8000h	23.6.10/543
1A2	INTC Priority Select Register (INTC_PSR161)	16	R/W	8000h	23.6.10/543
1A4	INTC Priority Select Register (INTC_PSR162)	16	R/W	8000h	23.6.10/543
1A6	INTC Priority Select Register (INTC_PSR163)	16	R/W	8000h	23.6.10/543
1A8	INTC Priority Select Register (INTC_PSR164)	16	R/W	8000h	23.6.10/543
1AA	INTC Priority Select Register (INTC_PSR165)	16	R/W	8000h	23.6.10/543
1AC	INTC Priority Select Register (INTC_PSR166)	16	R/W	8000h	23.6.10/543
1AE	INTC Priority Select Register (INTC_PSR167)	16	R/W	8000h	23.6.10/543
1B0	INTC Priority Select Register (INTC_PSR168)	16	R/W	8000h	23.6.10/543
1B2	INTC Priority Select Register (INTC_PSR169)	16	R/W	8000h	23.6.10/543
1B4	INTC Priority Select Register (INTC_PSR170)	16	R/W	8000h	23.6.10/543
1B6	INTC Priority Select Register (INTC_PSR171)	16	R/W	8000h	23.6.10/543
1B8	INTC Priority Select Register (INTC_PSR172)	16	R/W	8000h	23.6.10/543
1BA	INTC Priority Select Register (INTC_PSR173)	16	R/W	8000h	23.6.10/543
1BC	INTC Priority Select Register (INTC_PSR174)	16	R/W	8000h	23.6.10/543
1BE	INTC Priority Select Register (INTC_PSR175)	16	R/W	8000h	23.6.10/543
1C0	INTC Priority Select Register (INTC_PSR176)	16	R/W	8000h	23.6.10/543
1C2	INTC Priority Select Register (INTC_PSR177)	16	R/W	8000h	23.6.10/543
1C4	INTC Priority Select Register (INTC_PSR178)	16	R/W	8000h	23.6.10/543
1C6	INTC Priority Select Register (INTC_PSR179)	16	R/W	8000h	23.6.10/543
1C8	INTC Priority Select Register (INTC_PSR180)	16	R/W	8000h	23.6.10/543
1CA	INTC Priority Select Register (INTC_PSR181)	16	R/W	8000h	23.6.10/543
1CC	INTC Priority Select Register (INTC_PSR182)	16	R/W	8000h	23.6.10/543
1CE	INTC Priority Select Register (INTC_PSR183)	16	R/W	8000h	23.6.10/543
1D0	INTC Priority Select Register (INTC_PSR184)	16	R/W	8000h	23.6.10/543
1D2	INTC Priority Select Register (INTC_PSR185)	16	R/W	8000h	23.6.10/543
1D4	INTC Priority Select Register (INTC_PSR186)	16	R/W	8000h	23.6.10/543
1D6	INTC Priority Select Register (INTC_PSR187)	16	R/W	8000h	23.6.10/543
1D8	INTC Priority Select Register (INTC_PSR188)	16	R/W	8000h	23.6.10/543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1DA	INTC Priority Select Register (INTC_PSR189)	16	R/W	8000h	23.6.10/543
1DC	INTC Priority Select Register (INTC_PSR190)	16	R/W	8000h	23.6.10/543
1DE	INTC Priority Select Register (INTC_PSR191)	16	R/W	8000h	23.6.10/543
1E0	INTC Priority Select Register (INTC_PSR192)	16	R/W	8000h	23.6.10/543
1E2	INTC Priority Select Register (INTC_PSR193)	16	R/W	8000h	23.6.10/543
1E4	INTC Priority Select Register (INTC_PSR194)	16	R/W	8000h	23.6.10/543
1E6	INTC Priority Select Register (INTC_PSR195)	16	R/W	8000h	23.6.10/543
1E8	INTC Priority Select Register (INTC_PSR196)	16	R/W	8000h	23.6.10/543
1EA	INTC Priority Select Register (INTC_PSR197)	16	R/W	8000h	23.6.10/543
1EC	INTC Priority Select Register (INTC_PSR198)	16	R/W	8000h	23.6.10/543
1EE	INTC Priority Select Register (INTC_PSR199)	16	R/W	8000h	23.6.10/543
1F0	INTC Priority Select Register (INTC_PSR200)	16	R/W	8000h	23.6.10/543
1F2	INTC Priority Select Register (INTC_PSR201)	16	R/W	8000h	23.6.10/543
1F4	INTC Priority Select Register (INTC_PSR202)	16	R/W	8000h	23.6.10/543
1F6	INTC Priority Select Register (INTC_PSR203)	16	R/W	8000h	23.6.10/543
1F8	INTC Priority Select Register (INTC_PSR204)	16	R/W	8000h	23.6.10/543
1FA	INTC Priority Select Register (INTC_PSR205)	16	R/W	8000h	23.6.10/543
1FC	INTC Priority Select Register (INTC_PSR206)	16	R/W	8000h	23.6.10/543
1FE	INTC Priority Select Register (INTC_PSR207)	16	R/W	8000h	23.6.10/543
200	INTC Priority Select Register (INTC_PSR208)	16	R/W	8000h	23.6.10/543
202	INTC Priority Select Register (INTC_PSR209)	16	R/W	8000h	23.6.10/543
204	INTC Priority Select Register (INTC_PSR210)	16	R/W	8000h	23.6.10/543
206	INTC Priority Select Register (INTC_PSR211)	16	R/W	8000h	23.6.10/543
208	INTC Priority Select Register (INTC_PSR212)	16	R/W	8000h	23.6.10/543
20A	INTC Priority Select Register (INTC_PSR213)	16	R/W	8000h	23.6.10/543
20C	INTC Priority Select Register (INTC_PSR214)	16	R/W	8000h	23.6.10/543
20E	INTC Priority Select Register (INTC_PSR215)	16	R/W	8000h	23.6.10/543
210	INTC Priority Select Register (INTC_PSR216)	16	R/W	8000h	23.6.10/543
212	INTC Priority Select Register (INTC_PSR217)	16	R/W	8000h	23.6.10/543
214	INTC Priority Select Register (INTC_PSR218)	16	R/W	8000h	23.6.10/543
216	INTC Priority Select Register (INTC_PSR219)	16	R/W	8000h	23.6.10/543
218	INTC Priority Select Register (INTC_PSR220)	16	R/W	8000h	23.6.10/543
21A	INTC Priority Select Register (INTC_PSR221)	16	R/W	8000h	23.6.10/543
21C	INTC Priority Select Register (INTC_PSR222)	16	R/W	8000h	23.6.10/543
21E	INTC Priority Select Register (INTC_PSR223)	16	R/W	8000h	23.6.10/543
220	INTC Priority Select Register (INTC_PSR224)	16	R/W	8000h	23.6.10/543
222	INTC Priority Select Register (INTC_PSR225)	16	R/W	8000h	23.6.10/543
224	INTC Priority Select Register (INTC_PSR226)	16	R/W	8000h	23.6.10/543
226	INTC Priority Select Register (INTC_PSR227)	16	R/W	8000h	23.6.10/543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
228	INTC Priority Select Register (INTC_PSR228)	16	R/W	8000h	23.6.10/543
22A	INTC Priority Select Register (INTC_PSR229)	16	R/W	8000h	23.6.10/543
22C	INTC Priority Select Register (INTC_PSR230)	16	R/W	8000h	23.6.10/543
22E	INTC Priority Select Register (INTC_PSR231)	16	R/W	8000h	23.6.10/543
230	INTC Priority Select Register (INTC_PSR232)	16	R/W	8000h	23.6.10/543
232	INTC Priority Select Register (INTC_PSR233)	16	R/W	8000h	23.6.10/543
234	INTC Priority Select Register (INTC_PSR234)	16	R/W	8000h	23.6.10/543
236	INTC Priority Select Register (INTC_PSR235)	16	R/W	8000h	23.6.10/543
238	INTC Priority Select Register (INTC_PSR236)	16	R/W	8000h	23.6.10/543
23A	INTC Priority Select Register (INTC_PSR237)	16	R/W	8000h	23.6.10/543
23C	INTC Priority Select Register (INTC_PSR238)	16	R/W	8000h	23.6.10/543
23E	INTC Priority Select Register (INTC_PSR239)	16	R/W	8000h	23.6.10/543
240	INTC Priority Select Register (INTC_PSR240)	16	R/W	8000h	23.6.10/543
242	INTC Priority Select Register (INTC_PSR241)	16	R/W	8000h	23.6.10/543
244	INTC Priority Select Register (INTC_PSR242)	16	R/W	8000h	23.6.10/543
246	INTC Priority Select Register (INTC_PSR243)	16	R/W	8000h	23.6.10/543
248	INTC Priority Select Register (INTC_PSR244)	16	R/W	8000h	23.6.10/543
24A	INTC Priority Select Register (INTC_PSR245)	16	R/W	8000h	23.6.10/543
24C	INTC Priority Select Register (INTC_PSR246)	16	R/W	8000h	23.6.10/543
24E	INTC Priority Select Register (INTC_PSR247)	16	R/W	8000h	23.6.10/543
250	INTC Priority Select Register (INTC_PSR248)	16	R/W	8000h	23.6.10/543
252	INTC Priority Select Register (INTC_PSR249)	16	R/W	8000h	23.6.10/543
254	INTC Priority Select Register (INTC_PSR250)	16	R/W	8000h	23.6.10/543
256	INTC Priority Select Register (INTC_PSR251)	16	R/W	8000h	23.6.10/543
258	INTC Priority Select Register (INTC_PSR252)	16	R/W	8000h	23.6.10/543
25A	INTC Priority Select Register (INTC_PSR253)	16	R/W	8000h	23.6.10/543
25C	INTC Priority Select Register (INTC_PSR254)	16	R/W	8000h	23.6.10/543
25E	INTC Priority Select Register (INTC_PSR255)	16	R/W	8000h	23.6.10/543
260	INTC Priority Select Register (INTC_PSR256)	16	R/W	8000h	23.6.10/543
262	INTC Priority Select Register (INTC_PSR257)	16	R/W	8000h	23.6.10/543
264	INTC Priority Select Register (INTC_PSR258)	16	R/W	8000h	23.6.10/543
266	INTC Priority Select Register (INTC_PSR259)	16	R/W	8000h	23.6.10/543
268	INTC Priority Select Register (INTC_PSR260)	16	R/W	8000h	23.6.10/543
26A	INTC Priority Select Register (INTC_PSR261)	16	R/W	8000h	23.6.10/543
26C	INTC Priority Select Register (INTC_PSR262)	16	R/W	8000h	23.6.10/543
26E	INTC Priority Select Register (INTC_PSR263)	16	R/W	8000h	23.6.10/543
270	INTC Priority Select Register (INTC_PSR264)	16	R/W	8000h	23.6.10/543
272	INTC Priority Select Register (INTC_PSR265)	16	R/W	8000h	23.6.10/543
274	INTC Priority Select Register (INTC_PSR266)	16	R/W	8000h	23.6.10/543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
276	INTC Priority Select Register (INTC_PSR267)	16	R/W	8000h	23.6.10/543
278	INTC Priority Select Register (INTC_PSR268)	16	R/W	8000h	23.6.10/ 543
27A	INTC Priority Select Register (INTC_PSR269)	16	R/W	8000h	23.6.10/ 543
27C	INTC Priority Select Register (INTC_PSR270)	16	R/W	8000h	23.6.10/ 543
27E	INTC Priority Select Register (INTC_PSR271)	16	R/W	8000h	23.6.10/ 543
280	INTC Priority Select Register (INTC_PSR272)	16	R/W	8000h	23.6.10/ 543
282	INTC Priority Select Register (INTC_PSR273)	16	R/W	8000h	23.6.10/ 543
284	INTC Priority Select Register (INTC_PSR274)	16	R/W	8000h	23.6.10/ 543
286	INTC Priority Select Register (INTC_PSR275)	16	R/W	8000h	23.6.10/ 543
288	INTC Priority Select Register (INTC_PSR276)	16	R/W	8000h	23.6.10/ 543
28A	INTC Priority Select Register (INTC_PSR277)	16	R/W	8000h	23.6.10/ 543
28C	INTC Priority Select Register (INTC_PSR278)	16	R/W	8000h	23.6.10/ 543
28E	INTC Priority Select Register (INTC_PSR279)	16	R/W	8000h	23.6.10/ 543
290	INTC Priority Select Register (INTC_PSR280)	16	R/W	8000h	23.6.10/ 543
292	INTC Priority Select Register (INTC_PSR281)	16	R/W	8000h	23.6.10/ 543
294	INTC Priority Select Register (INTC_PSR282)	16	R/W	8000h	23.6.10/ 543
296	INTC Priority Select Register (INTC_PSR283)	16	R/W	8000h	23.6.10/ 543
298	INTC Priority Select Register (INTC_PSR284)	16	R/W	8000h	23.6.10/ 543
29A	INTC Priority Select Register (INTC_PSR285)	16	R/W	8000h	23.6.10/ 543
29C	INTC Priority Select Register (INTC_PSR286)	16	R/W	8000h	23.6.10/ 543
29E	INTC Priority Select Register (INTC_PSR287)	16	R/W	8000h	23.6.10/ 543
2A0	INTC Priority Select Register (INTC_PSR288)	16	R/W	8000h	23.6.10/ 543
2A2	INTC Priority Select Register (INTC_PSR289)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2A4	INTC Priority Select Register (INTC_PSR290)	16	R/W	8000h	23.6.10/ 543
2A6	INTC Priority Select Register (INTC_PSR291)	16	R/W	8000h	23.6.10/ 543
2A8	INTC Priority Select Register (INTC_PSR292)	16	R/W	8000h	23.6.10/ 543
2AA	INTC Priority Select Register (INTC_PSR293)	16	R/W	8000h	23.6.10/ 543
2AC	INTC Priority Select Register (INTC_PSR294)	16	R/W	8000h	23.6.10/ 543
2AE	INTC Priority Select Register (INTC_PSR295)	16	R/W	8000h	23.6.10/ 543
2B0	INTC Priority Select Register (INTC_PSR296)	16	R/W	8000h	23.6.10/ 543
2B2	INTC Priority Select Register (INTC_PSR297)	16	R/W	8000h	23.6.10/ 543
2B4	INTC Priority Select Register (INTC_PSR298)	16	R/W	8000h	23.6.10/ 543
2B6	INTC Priority Select Register (INTC_PSR299)	16	R/W	8000h	23.6.10/ 543
2B8	INTC Priority Select Register (INTC_PSR300)	16	R/W	8000h	23.6.10/ 543
2BA	INTC Priority Select Register (INTC_PSR301)	16	R/W	8000h	23.6.10/ 543
2BC	INTC Priority Select Register (INTC_PSR302)	16	R/W	8000h	23.6.10/ 543
2BE	INTC Priority Select Register (INTC_PSR303)	16	R/W	8000h	23.6.10/ 543
2C0	INTC Priority Select Register (INTC_PSR304)	16	R/W	8000h	23.6.10/ 543
2C2	INTC Priority Select Register (INTC_PSR305)	16	R/W	8000h	23.6.10/ 543
2C4	INTC Priority Select Register (INTC_PSR306)	16	R/W	8000h	23.6.10/ 543
2C6	INTC Priority Select Register (INTC_PSR307)	16	R/W	8000h	23.6.10/ 543
2C8	INTC Priority Select Register (INTC_PSR308)	16	R/W	8000h	23.6.10/ 543
2CA	INTC Priority Select Register (INTC_PSR309)	16	R/W	8000h	23.6.10/ 543
2CC	INTC Priority Select Register (INTC_PSR310)	16	R/W	8000h	23.6.10/ 543
2CE	INTC Priority Select Register (INTC_PSR311)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2D0	INTC Priority Select Register (INTC_PSR312)	16	R/W	8000h	23.6.10/ 543
2D2	INTC Priority Select Register (INTC_PSR313)	16	R/W	8000h	23.6.10/ 543
2D4	INTC Priority Select Register (INTC_PSR314)	16	R/W	8000h	23.6.10/ 543
2D6	INTC Priority Select Register (INTC_PSR315)	16	R/W	8000h	23.6.10/ 543
2D8	INTC Priority Select Register (INTC_PSR316)	16	R/W	8000h	23.6.10/ 543
2DA	INTC Priority Select Register (INTC_PSR317)	16	R/W	8000h	23.6.10/ 543
2DC	INTC Priority Select Register (INTC_PSR318)	16	R/W	8000h	23.6.10/ 543
2DE	INTC Priority Select Register (INTC_PSR319)	16	R/W	8000h	23.6.10/ 543
2E0	INTC Priority Select Register (INTC_PSR320)	16	R/W	8000h	23.6.10/ 543
2E2	INTC Priority Select Register (INTC_PSR321)	16	R/W	8000h	23.6.10/ 543
2E4	INTC Priority Select Register (INTC_PSR322)	16	R/W	8000h	23.6.10/ 543
2E6	INTC Priority Select Register (INTC_PSR323)	16	R/W	8000h	23.6.10/ 543
2E8	INTC Priority Select Register (INTC_PSR324)	16	R/W	8000h	23.6.10/ 543
2EA	INTC Priority Select Register (INTC_PSR325)	16	R/W	8000h	23.6.10/ 543
2EC	INTC Priority Select Register (INTC_PSR326)	16	R/W	8000h	23.6.10/ 543
2EE	INTC Priority Select Register (INTC_PSR327)	16	R/W	8000h	23.6.10/ 543
2F0	INTC Priority Select Register (INTC_PSR328)	16	R/W	8000h	23.6.10/ 543
2F2	INTC Priority Select Register (INTC_PSR329)	16	R/W	8000h	23.6.10/ 543
2F4	INTC Priority Select Register (INTC_PSR330)	16	R/W	8000h	23.6.10/ 543
2F6	INTC Priority Select Register (INTC_PSR331)	16	R/W	8000h	23.6.10/ 543
2F8	INTC Priority Select Register (INTC_PSR332)	16	R/W	8000h	23.6.10/ 543
2FA	INTC Priority Select Register (INTC_PSR333)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2FC	INTC Priority Select Register (INTC_PSR334)	16	R/W	8000h	23.6.10/ 543
2FE	INTC Priority Select Register (INTC_PSR335)	16	R/W	8000h	23.6.10/ 543
300	INTC Priority Select Register (INTC_PSR336)	16	R/W	8000h	23.6.10/ 543
302	INTC Priority Select Register (INTC_PSR337)	16	R/W	8000h	23.6.10/ 543
304	INTC Priority Select Register (INTC_PSR338)	16	R/W	8000h	23.6.10/ 543
306	INTC Priority Select Register (INTC_PSR339)	16	R/W	8000h	23.6.10/ 543
308	INTC Priority Select Register (INTC_PSR340)	16	R/W	8000h	23.6.10/ 543
30A	INTC Priority Select Register (INTC_PSR341)	16	R/W	8000h	23.6.10/ 543
30C	INTC Priority Select Register (INTC_PSR342)	16	R/W	8000h	23.6.10/ 543
30E	INTC Priority Select Register (INTC_PSR343)	16	R/W	8000h	23.6.10/ 543
310	INTC Priority Select Register (INTC_PSR344)	16	R/W	8000h	23.6.10/ 543
312	INTC Priority Select Register (INTC_PSR345)	16	R/W	8000h	23.6.10/ 543
314	INTC Priority Select Register (INTC_PSR346)	16	R/W	8000h	23.6.10/ 543
316	INTC Priority Select Register (INTC_PSR347)	16	R/W	8000h	23.6.10/ 543
318	INTC Priority Select Register (INTC_PSR348)	16	R/W	8000h	23.6.10/ 543
31A	INTC Priority Select Register (INTC_PSR349)	16	R/W	8000h	23.6.10/ 543
31C	INTC Priority Select Register (INTC_PSR350)	16	R/W	8000h	23.6.10/ 543
31E	INTC Priority Select Register (INTC_PSR351)	16	R/W	8000h	23.6.10/ 543
320	INTC Priority Select Register (INTC_PSR352)	16	R/W	8000h	23.6.10/ 543
322	INTC Priority Select Register (INTC_PSR353)	16	R/W	8000h	23.6.10/ 543
324	INTC Priority Select Register (INTC_PSR354)	16	R/W	8000h	23.6.10/ 543
326	INTC Priority Select Register (INTC_PSR355)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
328	INTC Priority Select Register (INTC_PSR356)	16	R/W	8000h	23.6.10/ 543
32A	INTC Priority Select Register (INTC_PSR357)	16	R/W	8000h	23.6.10/ 543
32C	INTC Priority Select Register (INTC_PSR358)	16	R/W	8000h	23.6.10/ 543
32E	INTC Priority Select Register (INTC_PSR359)	16	R/W	8000h	23.6.10/ 543
330	INTC Priority Select Register (INTC_PSR360)	16	R/W	8000h	23.6.10/ 543
332	INTC Priority Select Register (INTC_PSR361)	16	R/W	8000h	23.6.10/ 543
334	INTC Priority Select Register (INTC_PSR362)	16	R/W	8000h	23.6.10/ 543
336	INTC Priority Select Register (INTC_PSR363)	16	R/W	8000h	23.6.10/ 543
338	INTC Priority Select Register (INTC_PSR364)	16	R/W	8000h	23.6.10/ 543
33A	INTC Priority Select Register (INTC_PSR365)	16	R/W	8000h	23.6.10/ 543
33C	INTC Priority Select Register (INTC_PSR366)	16	R/W	8000h	23.6.10/ 543
33E	INTC Priority Select Register (INTC_PSR367)	16	R/W	8000h	23.6.10/ 543
340	INTC Priority Select Register (INTC_PSR368)	16	R/W	8000h	23.6.10/ 543
342	INTC Priority Select Register (INTC_PSR369)	16	R/W	8000h	23.6.10/ 543
344	INTC Priority Select Register (INTC_PSR370)	16	R/W	8000h	23.6.10/ 543
346	INTC Priority Select Register (INTC_PSR371)	16	R/W	8000h	23.6.10/ 543
348	INTC Priority Select Register (INTC_PSR372)	16	R/W	8000h	23.6.10/ 543
34A	INTC Priority Select Register (INTC_PSR373)	16	R/W	8000h	23.6.10/ 543
34C	INTC Priority Select Register (INTC_PSR374)	16	R/W	8000h	23.6.10/ 543
34E	INTC Priority Select Register (INTC_PSR375)	16	R/W	8000h	23.6.10/ 543
350	INTC Priority Select Register (INTC_PSR376)	16	R/W	8000h	23.6.10/ 543
352	INTC Priority Select Register (INTC_PSR377)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
354	INTC Priority Select Register (INTC_PSR378)	16	R/W	8000h	23.6.10/ 543
356	INTC Priority Select Register (INTC_PSR379)	16	R/W	8000h	23.6.10/ 543
358	INTC Priority Select Register (INTC_PSR380)	16	R/W	8000h	23.6.10/ 543
35A	INTC Priority Select Register (INTC_PSR381)	16	R/W	8000h	23.6.10/ 543
35C	INTC Priority Select Register (INTC_PSR382)	16	R/W	8000h	23.6.10/ 543
35E	INTC Priority Select Register (INTC_PSR383)	16	R/W	8000h	23.6.10/ 543
360	INTC Priority Select Register (INTC_PSR384)	16	R/W	8000h	23.6.10/ 543
362	INTC Priority Select Register (INTC_PSR385)	16	R/W	8000h	23.6.10/ 543
364	INTC Priority Select Register (INTC_PSR386)	16	R/W	8000h	23.6.10/ 543
366	INTC Priority Select Register (INTC_PSR387)	16	R/W	8000h	23.6.10/ 543
368	INTC Priority Select Register (INTC_PSR388)	16	R/W	8000h	23.6.10/ 543
36A	INTC Priority Select Register (INTC_PSR389)	16	R/W	8000h	23.6.10/ 543
36C	INTC Priority Select Register (INTC_PSR390)	16	R/W	8000h	23.6.10/ 543
36E	INTC Priority Select Register (INTC_PSR391)	16	R/W	8000h	23.6.10/ 543
370	INTC Priority Select Register (INTC_PSR392)	16	R/W	8000h	23.6.10/ 543
372	INTC Priority Select Register (INTC_PSR393)	16	R/W	8000h	23.6.10/ 543
374	INTC Priority Select Register (INTC_PSR394)	16	R/W	8000h	23.6.10/ 543
376	INTC Priority Select Register (INTC_PSR395)	16	R/W	8000h	23.6.10/ 543
378	INTC Priority Select Register (INTC_PSR396)	16	R/W	8000h	23.6.10/ 543
37A	INTC Priority Select Register (INTC_PSR397)	16	R/W	8000h	23.6.10/ 543
37C	INTC Priority Select Register (INTC_PSR398)	16	R/W	8000h	23.6.10/ 543
37E	INTC Priority Select Register (INTC_PSR399)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
380	INTC Priority Select Register (INTC_PSR400)	16	R/W	8000h	23.6.10/ 543
382	INTC Priority Select Register (INTC_PSR401)	16	R/W	8000h	23.6.10/ 543
384	INTC Priority Select Register (INTC_PSR402)	16	R/W	8000h	23.6.10/ 543
386	INTC Priority Select Register (INTC_PSR403)	16	R/W	8000h	23.6.10/ 543
388	INTC Priority Select Register (INTC_PSR404)	16	R/W	8000h	23.6.10/ 543
38A	INTC Priority Select Register (INTC_PSR405)	16	R/W	8000h	23.6.10/ 543
38C	INTC Priority Select Register (INTC_PSR406)	16	R/W	8000h	23.6.10/ 543
38E	INTC Priority Select Register (INTC_PSR407)	16	R/W	8000h	23.6.10/ 543
390	INTC Priority Select Register (INTC_PSR408)	16	R/W	8000h	23.6.10/ 543
392	INTC Priority Select Register (INTC_PSR409)	16	R/W	8000h	23.6.10/ 543
394	INTC Priority Select Register (INTC_PSR410)	16	R/W	8000h	23.6.10/ 543
396	INTC Priority Select Register (INTC_PSR411)	16	R/W	8000h	23.6.10/ 543
398	INTC Priority Select Register (INTC_PSR412)	16	R/W	8000h	23.6.10/ 543
39A	INTC Priority Select Register (INTC_PSR413)	16	R/W	8000h	23.6.10/ 543
39C	INTC Priority Select Register (INTC_PSR414)	16	R/W	8000h	23.6.10/ 543
39E	INTC Priority Select Register (INTC_PSR415)	16	R/W	8000h	23.6.10/ 543
3A0	INTC Priority Select Register (INTC_PSR416)	16	R/W	8000h	23.6.10/ 543
3A2	INTC Priority Select Register (INTC_PSR417)	16	R/W	8000h	23.6.10/ 543
3A4	INTC Priority Select Register (INTC_PSR418)	16	R/W	8000h	23.6.10/ 543
3A6	INTC Priority Select Register (INTC_PSR419)	16	R/W	8000h	23.6.10/ 543
3A8	INTC Priority Select Register (INTC_PSR420)	16	R/W	8000h	23.6.10/ 543
3AA	INTC Priority Select Register (INTC_PSR421)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3AC	INTC Priority Select Register (INTC_PSR422)	16	R/W	8000h	23.6.10/ 543
3AE	INTC Priority Select Register (INTC_PSR423)	16	R/W	8000h	23.6.10/ 543
3B0	INTC Priority Select Register (INTC_PSR424)	16	R/W	8000h	23.6.10/ 543
3B2	INTC Priority Select Register (INTC_PSR425)	16	R/W	8000h	23.6.10/ 543
3B4	INTC Priority Select Register (INTC_PSR426)	16	R/W	8000h	23.6.10/ 543
3B6	INTC Priority Select Register (INTC_PSR427)	16	R/W	8000h	23.6.10/ 543
3B8	INTC Priority Select Register (INTC_PSR428)	16	R/W	8000h	23.6.10/ 543
3BA	INTC Priority Select Register (INTC_PSR429)	16	R/W	8000h	23.6.10/ 543
3BC	INTC Priority Select Register (INTC_PSR430)	16	R/W	8000h	23.6.10/ 543
3BE	INTC Priority Select Register (INTC_PSR431)	16	R/W	8000h	23.6.10/ 543
3C0	INTC Priority Select Register (INTC_PSR432)	16	R/W	8000h	23.6.10/ 543
3C2	INTC Priority Select Register (INTC_PSR433)	16	R/W	8000h	23.6.10/ 543
3C4	INTC Priority Select Register (INTC_PSR434)	16	R/W	8000h	23.6.10/ 543
3C6	INTC Priority Select Register (INTC_PSR435)	16	R/W	8000h	23.6.10/ 543
3C8	INTC Priority Select Register (INTC_PSR436)	16	R/W	8000h	23.6.10/ 543
3CA	INTC Priority Select Register (INTC_PSR437)	16	R/W	8000h	23.6.10/ 543
3CC	INTC Priority Select Register (INTC_PSR438)	16	R/W	8000h	23.6.10/ 543
3CE	INTC Priority Select Register (INTC_PSR439)	16	R/W	8000h	23.6.10/ 543
3D0	INTC Priority Select Register (INTC_PSR440)	16	R/W	8000h	23.6.10/ 543
3D2	INTC Priority Select Register (INTC_PSR441)	16	R/W	8000h	23.6.10/ 543
3D4	INTC Priority Select Register (INTC_PSR442)	16	R/W	8000h	23.6.10/ 543
3D6	INTC Priority Select Register (INTC_PSR443)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3D8	INTC Priority Select Register (INTC_PSR444)	16	R/W	8000h	23.6.10/ 543
3DA	INTC Priority Select Register (INTC_PSR445)	16	R/W	8000h	23.6.10/ 543
3DC	INTC Priority Select Register (INTC_PSR446)	16	R/W	8000h	23.6.10/ 543
3DE	INTC Priority Select Register (INTC_PSR447)	16	R/W	8000h	23.6.10/ 543
3E0	INTC Priority Select Register (INTC_PSR448)	16	R/W	8000h	23.6.10/ 543
3E2	INTC Priority Select Register (INTC_PSR449)	16	R/W	8000h	23.6.10/ 543
3E4	INTC Priority Select Register (INTC_PSR450)	16	R/W	8000h	23.6.10/ 543
3E6	INTC Priority Select Register (INTC_PSR451)	16	R/W	8000h	23.6.10/ 543
3E8	INTC Priority Select Register (INTC_PSR452)	16	R/W	8000h	23.6.10/ 543
3EA	INTC Priority Select Register (INTC_PSR453)	16	R/W	8000h	23.6.10/ 543
3EC	INTC Priority Select Register (INTC_PSR454)	16	R/W	8000h	23.6.10/ 543
3EE	INTC Priority Select Register (INTC_PSR455)	16	R/W	8000h	23.6.10/ 543
3F0	INTC Priority Select Register (INTC_PSR456)	16	R/W	8000h	23.6.10/ 543
3F2	INTC Priority Select Register (INTC_PSR457)	16	R/W	8000h	23.6.10/ 543
3F4	INTC Priority Select Register (INTC_PSR458)	16	R/W	8000h	23.6.10/ 543
3F6	INTC Priority Select Register (INTC_PSR459)	16	R/W	8000h	23.6.10/ 543
3F8	INTC Priority Select Register (INTC_PSR460)	16	R/W	8000h	23.6.10/ 543
3FA	INTC Priority Select Register (INTC_PSR461)	16	R/W	8000h	23.6.10/ 543
3FC	INTC Priority Select Register (INTC_PSR462)	16	R/W	8000h	23.6.10/ 543
3FE	INTC Priority Select Register (INTC_PSR463)	16	R/W	8000h	23.6.10/ 543
400	INTC Priority Select Register (INTC_PSR464)	16	R/W	8000h	23.6.10/ 543
402	INTC Priority Select Register (INTC_PSR465)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
404	INTC Priority Select Register (INTC_PSR466)	16	R/W	8000h	23.6.10/ 543
406	INTC Priority Select Register (INTC_PSR467)	16	R/W	8000h	23.6.10/ 543
408	INTC Priority Select Register (INTC_PSR468)	16	R/W	8000h	23.6.10/ 543
40A	INTC Priority Select Register (INTC_PSR469)	16	R/W	8000h	23.6.10/ 543
40C	INTC Priority Select Register (INTC_PSR470)	16	R/W	8000h	23.6.10/ 543
40E	INTC Priority Select Register (INTC_PSR471)	16	R/W	8000h	23.6.10/ 543
410	INTC Priority Select Register (INTC_PSR472)	16	R/W	8000h	23.6.10/ 543
412	INTC Priority Select Register (INTC_PSR473)	16	R/W	8000h	23.6.10/ 543
414	INTC Priority Select Register (INTC_PSR474)	16	R/W	8000h	23.6.10/ 543
416	INTC Priority Select Register (INTC_PSR475)	16	R/W	8000h	23.6.10/ 543
418	INTC Priority Select Register (INTC_PSR476)	16	R/W	8000h	23.6.10/ 543
41A	INTC Priority Select Register (INTC_PSR477)	16	R/W	8000h	23.6.10/ 543
41C	INTC Priority Select Register (INTC_PSR478)	16	R/W	8000h	23.6.10/ 543
41E	INTC Priority Select Register (INTC_PSR479)	16	R/W	8000h	23.6.10/ 543
420	INTC Priority Select Register (INTC_PSR480)	16	R/W	8000h	23.6.10/ 543
422	INTC Priority Select Register (INTC_PSR481)	16	R/W	8000h	23.6.10/ 543
424	INTC Priority Select Register (INTC_PSR482)	16	R/W	8000h	23.6.10/ 543
426	INTC Priority Select Register (INTC_PSR483)	16	R/W	8000h	23.6.10/ 543
428	INTC Priority Select Register (INTC_PSR484)	16	R/W	8000h	23.6.10/ 543
42A	INTC Priority Select Register (INTC_PSR485)	16	R/W	8000h	23.6.10/ 543
42C	INTC Priority Select Register (INTC_PSR486)	16	R/W	8000h	23.6.10/ 543
42E	INTC Priority Select Register (INTC_PSR487)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
430	INTC Priority Select Register (INTC_PSR488)	16	R/W	8000h	23.6.10/ 543
432	INTC Priority Select Register (INTC_PSR489)	16	R/W	8000h	23.6.10/ 543
434	INTC Priority Select Register (INTC_PSR490)	16	R/W	8000h	23.6.10/ 543
436	INTC Priority Select Register (INTC_PSR491)	16	R/W	8000h	23.6.10/ 543
438	INTC Priority Select Register (INTC_PSR492)	16	R/W	8000h	23.6.10/ 543
43A	INTC Priority Select Register (INTC_PSR493)	16	R/W	8000h	23.6.10/ 543
43C	INTC Priority Select Register (INTC_PSR494)	16	R/W	8000h	23.6.10/ 543
43E	INTC Priority Select Register (INTC_PSR495)	16	R/W	8000h	23.6.10/ 543
440	INTC Priority Select Register (INTC_PSR496)	16	R/W	8000h	23.6.10/ 543
442	INTC Priority Select Register (INTC_PSR497)	16	R/W	8000h	23.6.10/ 543
444	INTC Priority Select Register (INTC_PSR498)	16	R/W	8000h	23.6.10/ 543
446	INTC Priority Select Register (INTC_PSR499)	16	R/W	8000h	23.6.10/ 543
448	INTC Priority Select Register (INTC_PSR500)	16	R/W	8000h	23.6.10/ 543
44A	INTC Priority Select Register (INTC_PSR501)	16	R/W	8000h	23.6.10/ 543
44C	INTC Priority Select Register (INTC_PSR502)	16	R/W	8000h	23.6.10/ 543
44E	INTC Priority Select Register (INTC_PSR503)	16	R/W	8000h	23.6.10/ 543
450	INTC Priority Select Register (INTC_PSR504)	16	R/W	8000h	23.6.10/ 543
452	INTC Priority Select Register (INTC_PSR505)	16	R/W	8000h	23.6.10/ 543
454	INTC Priority Select Register (INTC_PSR506)	16	R/W	8000h	23.6.10/ 543
456	INTC Priority Select Register (INTC_PSR507)	16	R/W	8000h	23.6.10/ 543
458	INTC Priority Select Register (INTC_PSR508)	16	R/W	8000h	23.6.10/ 543
45A	INTC Priority Select Register (INTC_PSR509)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
45C	INTC Priority Select Register (INTC_PSR510)	16	R/W	8000h	23.6.10/ 543
45E	INTC Priority Select Register (INTC_PSR511)	16	R/W	8000h	23.6.10/ 543
460	INTC Priority Select Register (INTC_PSR512)	16	R/W	8000h	23.6.10/ 543
462	INTC Priority Select Register (INTC_PSR513)	16	R/W	8000h	23.6.10/ 543
464	INTC Priority Select Register (INTC_PSR514)	16	R/W	8000h	23.6.10/ 543
466	INTC Priority Select Register (INTC_PSR515)	16	R/W	8000h	23.6.10/ 543
468	INTC Priority Select Register (INTC_PSR516)	16	R/W	8000h	23.6.10/ 543
46A	INTC Priority Select Register (INTC_PSR517)	16	R/W	8000h	23.6.10/ 543
46C	INTC Priority Select Register (INTC_PSR518)	16	R/W	8000h	23.6.10/ 543
46E	INTC Priority Select Register (INTC_PSR519)	16	R/W	8000h	23.6.10/ 543
470	INTC Priority Select Register (INTC_PSR520)	16	R/W	8000h	23.6.10/ 543
472	INTC Priority Select Register (INTC_PSR521)	16	R/W	8000h	23.6.10/ 543
474	INTC Priority Select Register (INTC_PSR522)	16	R/W	8000h	23.6.10/ 543
476	INTC Priority Select Register (INTC_PSR523)	16	R/W	8000h	23.6.10/ 543
478	INTC Priority Select Register (INTC_PSR524)	16	R/W	8000h	23.6.10/ 543
47A	INTC Priority Select Register (INTC_PSR525)	16	R/W	8000h	23.6.10/ 543
47C	INTC Priority Select Register (INTC_PSR526)	16	R/W	8000h	23.6.10/ 543
47E	INTC Priority Select Register (INTC_PSR527)	16	R/W	8000h	23.6.10/ 543
480	INTC Priority Select Register (INTC_PSR528)	16	R/W	8000h	23.6.10/ 543
482	INTC Priority Select Register (INTC_PSR529)	16	R/W	8000h	23.6.10/ 543
484	INTC Priority Select Register (INTC_PSR530)	16	R/W	8000h	23.6.10/ 543
486	INTC Priority Select Register (INTC_PSR531)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
488	INTC Priority Select Register (INTC_PSR532)	16	R/W	8000h	23.6.10/ 543
48A	INTC Priority Select Register (INTC_PSR533)	16	R/W	8000h	23.6.10/ 543
48C	INTC Priority Select Register (INTC_PSR534)	16	R/W	8000h	23.6.10/ 543
48E	INTC Priority Select Register (INTC_PSR535)	16	R/W	8000h	23.6.10/ 543
490	INTC Priority Select Register (INTC_PSR536)	16	R/W	8000h	23.6.10/ 543
492	INTC Priority Select Register (INTC_PSR537)	16	R/W	8000h	23.6.10/ 543
494	INTC Priority Select Register (INTC_PSR538)	16	R/W	8000h	23.6.10/ 543
496	INTC Priority Select Register (INTC_PSR539)	16	R/W	8000h	23.6.10/ 543
498	INTC Priority Select Register (INTC_PSR540)	16	R/W	8000h	23.6.10/ 543
49A	INTC Priority Select Register (INTC_PSR541)	16	R/W	8000h	23.6.10/ 543
49C	INTC Priority Select Register (INTC_PSR542)	16	R/W	8000h	23.6.10/ 543
49E	INTC Priority Select Register (INTC_PSR543)	16	R/W	8000h	23.6.10/ 543
4A0	INTC Priority Select Register (INTC_PSR544)	16	R/W	8000h	23.6.10/ 543
4A2	INTC Priority Select Register (INTC_PSR545)	16	R/W	8000h	23.6.10/ 543
4A4	INTC Priority Select Register (INTC_PSR546)	16	R/W	8000h	23.6.10/ 543
4A6	INTC Priority Select Register (INTC_PSR547)	16	R/W	8000h	23.6.10/ 543
4A8	INTC Priority Select Register (INTC_PSR548)	16	R/W	8000h	23.6.10/ 543
4AA	INTC Priority Select Register (INTC_PSR549)	16	R/W	8000h	23.6.10/ 543
4AC	INTC Priority Select Register (INTC_PSR550)	16	R/W	8000h	23.6.10/ 543
4AE	INTC Priority Select Register (INTC_PSR551)	16	R/W	8000h	23.6.10/ 543
4B0	INTC Priority Select Register (INTC_PSR552)	16	R/W	8000h	23.6.10/ 543
4B2	INTC Priority Select Register (INTC_PSR553)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4B4	INTC Priority Select Register (INTC_PSR554)	16	R/W	8000h	23.6.10/ 543
4B6	INTC Priority Select Register (INTC_PSR555)	16	R/W	8000h	23.6.10/ 543
4B8	INTC Priority Select Register (INTC_PSR556)	16	R/W	8000h	23.6.10/ 543
4BA	INTC Priority Select Register (INTC_PSR557)	16	R/W	8000h	23.6.10/ 543
4BC	INTC Priority Select Register (INTC_PSR558)	16	R/W	8000h	23.6.10/ 543
4BE	INTC Priority Select Register (INTC_PSR559)	16	R/W	8000h	23.6.10/ 543
4C0	INTC Priority Select Register (INTC_PSR560)	16	R/W	8000h	23.6.10/ 543
4C2	INTC Priority Select Register (INTC_PSR561)	16	R/W	8000h	23.6.10/ 543
4C4	INTC Priority Select Register (INTC_PSR562)	16	R/W	8000h	23.6.10/ 543
4C6	INTC Priority Select Register (INTC_PSR563)	16	R/W	8000h	23.6.10/ 543
4C8	INTC Priority Select Register (INTC_PSR564)	16	R/W	8000h	23.6.10/ 543
4CA	INTC Priority Select Register (INTC_PSR565)	16	R/W	8000h	23.6.10/ 543
4CC	INTC Priority Select Register (INTC_PSR566)	16	R/W	8000h	23.6.10/ 543
4CE	INTC Priority Select Register (INTC_PSR567)	16	R/W	8000h	23.6.10/ 543
4D0	INTC Priority Select Register (INTC_PSR568)	16	R/W	8000h	23.6.10/ 543
4D2	INTC Priority Select Register (INTC_PSR569)	16	R/W	8000h	23.6.10/ 543
4D4	INTC Priority Select Register (INTC_PSR570)	16	R/W	8000h	23.6.10/ 543
4D6	INTC Priority Select Register (INTC_PSR571)	16	R/W	8000h	23.6.10/ 543
4D8	INTC Priority Select Register (INTC_PSR572)	16	R/W	8000h	23.6.10/ 543
4DA	INTC Priority Select Register (INTC_PSR573)	16	R/W	8000h	23.6.10/ 543
4DC	INTC Priority Select Register (INTC_PSR574)	16	R/W	8000h	23.6.10/ 543
4DE	INTC Priority Select Register (INTC_PSR575)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4E0	INTC Priority Select Register (INTC_PSR576)	16	R/W	8000h	23.6.10/ 543
4E2	INTC Priority Select Register (INTC_PSR577)	16	R/W	8000h	23.6.10/ 543
4E4	INTC Priority Select Register (INTC_PSR578)	16	R/W	8000h	23.6.10/ 543
4E6	INTC Priority Select Register (INTC_PSR579)	16	R/W	8000h	23.6.10/ 543
4E8	INTC Priority Select Register (INTC_PSR580)	16	R/W	8000h	23.6.10/ 543
4EA	INTC Priority Select Register (INTC_PSR581)	16	R/W	8000h	23.6.10/ 543
4EC	INTC Priority Select Register (INTC_PSR582)	16	R/W	8000h	23.6.10/ 543
4EE	INTC Priority Select Register (INTC_PSR583)	16	R/W	8000h	23.6.10/ 543
4F0	INTC Priority Select Register (INTC_PSR584)	16	R/W	8000h	23.6.10/ 543
4F2	INTC Priority Select Register (INTC_PSR585)	16	R/W	8000h	23.6.10/ 543
4F4	INTC Priority Select Register (INTC_PSR586)	16	R/W	8000h	23.6.10/ 543
4F6	INTC Priority Select Register (INTC_PSR587)	16	R/W	8000h	23.6.10/ 543
4F8	INTC Priority Select Register (INTC_PSR588)	16	R/W	8000h	23.6.10/ 543
4FA	INTC Priority Select Register (INTC_PSR589)	16	R/W	8000h	23.6.10/ 543
4FC	INTC Priority Select Register (INTC_PSR590)	16	R/W	8000h	23.6.10/ 543
4FE	INTC Priority Select Register (INTC_PSR591)	16	R/W	8000h	23.6.10/ 543
500	INTC Priority Select Register (INTC_PSR592)	16	R/W	8000h	23.6.10/ 543
502	INTC Priority Select Register (INTC_PSR593)	16	R/W	8000h	23.6.10/ 543
504	INTC Priority Select Register (INTC_PSR594)	16	R/W	8000h	23.6.10/ 543
506	INTC Priority Select Register (INTC_PSR595)	16	R/W	8000h	23.6.10/ 543
508	INTC Priority Select Register (INTC_PSR596)	16	R/W	8000h	23.6.10/ 543
50A	INTC Priority Select Register (INTC_PSR597)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
50C	INTC Priority Select Register (INTC_PSR598)	16	R/W	8000h	23.6.10/ 543
50E	INTC Priority Select Register (INTC_PSR599)	16	R/W	8000h	23.6.10/ 543
510	INTC Priority Select Register (INTC_PSR600)	16	R/W	8000h	23.6.10/ 543
512	INTC Priority Select Register (INTC_PSR601)	16	R/W	8000h	23.6.10/ 543
514	INTC Priority Select Register (INTC_PSR602)	16	R/W	8000h	23.6.10/ 543
516	INTC Priority Select Register (INTC_PSR603)	16	R/W	8000h	23.6.10/ 543
518	INTC Priority Select Register (INTC_PSR604)	16	R/W	8000h	23.6.10/ 543
51A	INTC Priority Select Register (INTC_PSR605)	16	R/W	8000h	23.6.10/ 543
51C	INTC Priority Select Register (INTC_PSR606)	16	R/W	8000h	23.6.10/ 543
51E	INTC Priority Select Register (INTC_PSR607)	16	R/W	8000h	23.6.10/ 543
520	INTC Priority Select Register (INTC_PSR608)	16	R/W	8000h	23.6.10/ 543
522	INTC Priority Select Register (INTC_PSR609)	16	R/W	8000h	23.6.10/ 543
524	INTC Priority Select Register (INTC_PSR610)	16	R/W	8000h	23.6.10/ 543
526	INTC Priority Select Register (INTC_PSR611)	16	R/W	8000h	23.6.10/ 543
528	INTC Priority Select Register (INTC_PSR612)	16	R/W	8000h	23.6.10/ 543
52A	INTC Priority Select Register (INTC_PSR613)	16	R/W	8000h	23.6.10/ 543
52C	INTC Priority Select Register (INTC_PSR614)	16	R/W	8000h	23.6.10/ 543
52E	INTC Priority Select Register (INTC_PSR615)	16	R/W	8000h	23.6.10/ 543
530	INTC Priority Select Register (INTC_PSR616)	16	R/W	8000h	23.6.10/ 543
532	INTC Priority Select Register (INTC_PSR617)	16	R/W	8000h	23.6.10/ 543
534	INTC Priority Select Register (INTC_PSR618)	16	R/W	8000h	23.6.10/ 543
536	INTC Priority Select Register (INTC_PSR619)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
538	INTC Priority Select Register (INTC_PSR620)	16	R/W	8000h	23.6.10/ 543
53A	INTC Priority Select Register (INTC_PSR621)	16	R/W	8000h	23.6.10/ 543
53C	INTC Priority Select Register (INTC_PSR622)	16	R/W	8000h	23.6.10/ 543
53E	INTC Priority Select Register (INTC_PSR623)	16	R/W	8000h	23.6.10/ 543
540	INTC Priority Select Register (INTC_PSR624)	16	R/W	8000h	23.6.10/ 543
542	INTC Priority Select Register (INTC_PSR625)	16	R/W	8000h	23.6.10/ 543
544	INTC Priority Select Register (INTC_PSR626)	16	R/W	8000h	23.6.10/ 543
546	INTC Priority Select Register (INTC_PSR627)	16	R/W	8000h	23.6.10/ 543
548	INTC Priority Select Register (INTC_PSR628)	16	R/W	8000h	23.6.10/ 543
54A	INTC Priority Select Register (INTC_PSR629)	16	R/W	8000h	23.6.10/ 543
54C	INTC Priority Select Register (INTC_PSR630)	16	R/W	8000h	23.6.10/ 543
54E	INTC Priority Select Register (INTC_PSR631)	16	R/W	8000h	23.6.10/ 543
550	INTC Priority Select Register (INTC_PSR632)	16	R/W	8000h	23.6.10/ 543
552	INTC Priority Select Register (INTC_PSR633)	16	R/W	8000h	23.6.10/ 543
554	INTC Priority Select Register (INTC_PSR634)	16	R/W	8000h	23.6.10/ 543
556	INTC Priority Select Register (INTC_PSR635)	16	R/W	8000h	23.6.10/ 543
558	INTC Priority Select Register (INTC_PSR636)	16	R/W	8000h	23.6.10/ 543
55A	INTC Priority Select Register (INTC_PSR637)	16	R/W	8000h	23.6.10/ 543
55C	INTC Priority Select Register (INTC_PSR638)	16	R/W	8000h	23.6.10/ 543
55E	INTC Priority Select Register (INTC_PSR639)	16	R/W	8000h	23.6.10/ 543
560	INTC Priority Select Register (INTC_PSR640)	16	R/W	8000h	23.6.10/ 543
562	INTC Priority Select Register (INTC_PSR641)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
564	INTC Priority Select Register (INTC_PSR642)	16	R/W	8000h	23.6.10/ 543
566	INTC Priority Select Register (INTC_PSR643)	16	R/W	8000h	23.6.10/ 543
568	INTC Priority Select Register (INTC_PSR644)	16	R/W	8000h	23.6.10/ 543
56A	INTC Priority Select Register (INTC_PSR645)	16	R/W	8000h	23.6.10/ 543
56C	INTC Priority Select Register (INTC_PSR646)	16	R/W	8000h	23.6.10/ 543
56E	INTC Priority Select Register (INTC_PSR647)	16	R/W	8000h	23.6.10/ 543
570	INTC Priority Select Register (INTC_PSR648)	16	R/W	8000h	23.6.10/ 543
572	INTC Priority Select Register (INTC_PSR649)	16	R/W	8000h	23.6.10/ 543
574	INTC Priority Select Register (INTC_PSR650)	16	R/W	8000h	23.6.10/ 543
576	INTC Priority Select Register (INTC_PSR651)	16	R/W	8000h	23.6.10/ 543
578	INTC Priority Select Register (INTC_PSR652)	16	R/W	8000h	23.6.10/ 543
57A	INTC Priority Select Register (INTC_PSR653)	16	R/W	8000h	23.6.10/ 543
57C	INTC Priority Select Register (INTC_PSR654)	16	R/W	8000h	23.6.10/ 543
57E	INTC Priority Select Register (INTC_PSR655)	16	R/W	8000h	23.6.10/ 543
580	INTC Priority Select Register (INTC_PSR656)	16	R/W	8000h	23.6.10/ 543
582	INTC Priority Select Register (INTC_PSR657)	16	R/W	8000h	23.6.10/ 543
584	INTC Priority Select Register (INTC_PSR658)	16	R/W	8000h	23.6.10/ 543
586	INTC Priority Select Register (INTC_PSR659)	16	R/W	8000h	23.6.10/ 543
588	INTC Priority Select Register (INTC_PSR660)	16	R/W	8000h	23.6.10/ 543
58A	INTC Priority Select Register (INTC_PSR661)	16	R/W	8000h	23.6.10/ 543
58C	INTC Priority Select Register (INTC_PSR662)	16	R/W	8000h	23.6.10/ 543
58E	INTC Priority Select Register (INTC_PSR663)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
590	INTC Priority Select Register (INTC_PSR664)	16	R/W	8000h	23.6.10/ 543
592	INTC Priority Select Register (INTC_PSR665)	16	R/W	8000h	23.6.10/ 543
594	INTC Priority Select Register (INTC_PSR666)	16	R/W	8000h	23.6.10/ 543
596	INTC Priority Select Register (INTC_PSR667)	16	R/W	8000h	23.6.10/ 543
598	INTC Priority Select Register (INTC_PSR668)	16	R/W	8000h	23.6.10/ 543
59A	INTC Priority Select Register (INTC_PSR669)	16	R/W	8000h	23.6.10/ 543
59C	INTC Priority Select Register (INTC_PSR670)	16	R/W	8000h	23.6.10/ 543
59E	INTC Priority Select Register (INTC_PSR671)	16	R/W	8000h	23.6.10/ 543
5A0	INTC Priority Select Register (INTC_PSR672)	16	R/W	8000h	23.6.10/ 543
5A2	INTC Priority Select Register (INTC_PSR673)	16	R/W	8000h	23.6.10/ 543
5A4	INTC Priority Select Register (INTC_PSR674)	16	R/W	8000h	23.6.10/ 543
5A6	INTC Priority Select Register (INTC_PSR675)	16	R/W	8000h	23.6.10/ 543
5A8	INTC Priority Select Register (INTC_PSR676)	16	R/W	8000h	23.6.10/ 543
5AA	INTC Priority Select Register (INTC_PSR677)	16	R/W	8000h	23.6.10/ 543
5AC	INTC Priority Select Register (INTC_PSR678)	16	R/W	8000h	23.6.10/ 543
5AE	INTC Priority Select Register (INTC_PSR679)	16	R/W	8000h	23.6.10/ 543
5B0	INTC Priority Select Register (INTC_PSR680)	16	R/W	8000h	23.6.10/ 543
5B2	INTC Priority Select Register (INTC_PSR681)	16	R/W	8000h	23.6.10/ 543
5B4	INTC Priority Select Register (INTC_PSR682)	16	R/W	8000h	23.6.10/ 543
5B6	INTC Priority Select Register (INTC_PSR683)	16	R/W	8000h	23.6.10/ 543
5B8	INTC Priority Select Register (INTC_PSR684)	16	R/W	8000h	23.6.10/ 543
5BA	INTC Priority Select Register (INTC_PSR685)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5BC	INTC Priority Select Register (INTC_PSR686)	16	R/W	8000h	23.6.10/ 543
5BE	INTC Priority Select Register (INTC_PSR687)	16	R/W	8000h	23.6.10/ 543
5C0	INTC Priority Select Register (INTC_PSR688)	16	R/W	8000h	23.6.10/ 543
5C2	INTC Priority Select Register (INTC_PSR689)	16	R/W	8000h	23.6.10/ 543
5C4	INTC Priority Select Register (INTC_PSR690)	16	R/W	8000h	23.6.10/ 543
5C6	INTC Priority Select Register (INTC_PSR691)	16	R/W	8000h	23.6.10/ 543
5C8	INTC Priority Select Register (INTC_PSR692)	16	R/W	8000h	23.6.10/ 543
5CA	INTC Priority Select Register (INTC_PSR693)	16	R/W	8000h	23.6.10/ 543
5CC	INTC Priority Select Register (INTC_PSR694)	16	R/W	8000h	23.6.10/ 543
5CE	INTC Priority Select Register (INTC_PSR695)	16	R/W	8000h	23.6.10/ 543
5D0	INTC Priority Select Register (INTC_PSR696)	16	R/W	8000h	23.6.10/ 543
5D2	INTC Priority Select Register (INTC_PSR697)	16	R/W	8000h	23.6.10/ 543
5D4	INTC Priority Select Register (INTC_PSR698)	16	R/W	8000h	23.6.10/ 543
5D6	INTC Priority Select Register (INTC_PSR699)	16	R/W	8000h	23.6.10/ 543
5D8	INTC Priority Select Register (INTC_PSR700)	16	R/W	8000h	23.6.10/ 543
5DA	INTC Priority Select Register (INTC_PSR701)	16	R/W	8000h	23.6.10/ 543
5DC	INTC Priority Select Register (INTC_PSR702)	16	R/W	8000h	23.6.10/ 543
5DE	INTC Priority Select Register (INTC_PSR703)	16	R/W	8000h	23.6.10/ 543
5E0	INTC Priority Select Register (INTC_PSR704)	16	R/W	8000h	23.6.10/ 543
5E2	INTC Priority Select Register (INTC_PSR705)	16	R/W	8000h	23.6.10/ 543
5E4	INTC Priority Select Register (INTC_PSR706)	16	R/W	8000h	23.6.10/ 543
5E6	INTC Priority Select Register (INTC_PSR707)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
5E8	INTC Priority Select Register (INTC_PSR708)	16	R/W	8000h	23.6.10/ 543
5EA	INTC Priority Select Register (INTC_PSR709)	16	R/W	8000h	23.6.10/ 543
5EC	INTC Priority Select Register (INTC_PSR710)	16	R/W	8000h	23.6.10/ 543
5EE	INTC Priority Select Register (INTC_PSR711)	16	R/W	8000h	23.6.10/ 543
5F0	INTC Priority Select Register (INTC_PSR712)	16	R/W	8000h	23.6.10/ 543
5F2	INTC Priority Select Register (INTC_PSR713)	16	R/W	8000h	23.6.10/ 543
5F4	INTC Priority Select Register (INTC_PSR714)	16	R/W	8000h	23.6.10/ 543
5F6	INTC Priority Select Register (INTC_PSR715)	16	R/W	8000h	23.6.10/ 543
5F8	INTC Priority Select Register (INTC_PSR716)	16	R/W	8000h	23.6.10/ 543
5FA	INTC Priority Select Register (INTC_PSR717)	16	R/W	8000h	23.6.10/ 543
5FC	INTC Priority Select Register (INTC_PSR718)	16	R/W	8000h	23.6.10/ 543
5FE	INTC Priority Select Register (INTC_PSR719)	16	R/W	8000h	23.6.10/ 543
600	INTC Priority Select Register (INTC_PSR720)	16	R/W	8000h	23.6.10/ 543
602	INTC Priority Select Register (INTC_PSR721)	16	R/W	8000h	23.6.10/ 543
604	INTC Priority Select Register (INTC_PSR722)	16	R/W	8000h	23.6.10/ 543
606	INTC Priority Select Register (INTC_PSR723)	16	R/W	8000h	23.6.10/ 543
608	INTC Priority Select Register (INTC_PSR724)	16	R/W	8000h	23.6.10/ 543
60A	INTC Priority Select Register (INTC_PSR725)	16	R/W	8000h	23.6.10/ 543
60C	INTC Priority Select Register (INTC_PSR726)	16	R/W	8000h	23.6.10/ 543
60E	INTC Priority Select Register (INTC_PSR727)	16	R/W	8000h	23.6.10/ 543
610	INTC Priority Select Register (INTC_PSR728)	16	R/W	8000h	23.6.10/ 543
612	INTC Priority Select Register (INTC_PSR729)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
614	INTC Priority Select Register (INTC_PSR730)	16	R/W	8000h	23.6.10/ 543
616	INTC Priority Select Register (INTC_PSR731)	16	R/W	8000h	23.6.10/ 543
618	INTC Priority Select Register (INTC_PSR732)	16	R/W	8000h	23.6.10/ 543
61A	INTC Priority Select Register (INTC_PSR733)	16	R/W	8000h	23.6.10/ 543
61C	INTC Priority Select Register (INTC_PSR734)	16	R/W	8000h	23.6.10/ 543
61E	INTC Priority Select Register (INTC_PSR735)	16	R/W	8000h	23.6.10/ 543
620	INTC Priority Select Register (INTC_PSR736)	16	R/W	8000h	23.6.10/ 543
622	INTC Priority Select Register (INTC_PSR737)	16	R/W	8000h	23.6.10/ 543
624	INTC Priority Select Register (INTC_PSR738)	16	R/W	8000h	23.6.10/ 543
626	INTC Priority Select Register (INTC_PSR739)	16	R/W	8000h	23.6.10/ 543
628	INTC Priority Select Register (INTC_PSR740)	16	R/W	8000h	23.6.10/ 543
62A	INTC Priority Select Register (INTC_PSR741)	16	R/W	8000h	23.6.10/ 543
62C	INTC Priority Select Register (INTC_PSR742)	16	R/W	8000h	23.6.10/ 543
62E	INTC Priority Select Register (INTC_PSR743)	16	R/W	8000h	23.6.10/ 543
630	INTC Priority Select Register (INTC_PSR744)	16	R/W	8000h	23.6.10/ 543
632	INTC Priority Select Register (INTC_PSR745)	16	R/W	8000h	23.6.10/ 543
634	INTC Priority Select Register (INTC_PSR746)	16	R/W	8000h	23.6.10/ 543
636	INTC Priority Select Register (INTC_PSR747)	16	R/W	8000h	23.6.10/ 543
638	INTC Priority Select Register (INTC_PSR748)	16	R/W	8000h	23.6.10/ 543
63A	INTC Priority Select Register (INTC_PSR749)	16	R/W	8000h	23.6.10/ 543
63C	INTC Priority Select Register (INTC_PSR750)	16	R/W	8000h	23.6.10/ 543
63E	INTC Priority Select Register (INTC_PSR751)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
640	INTC Priority Select Register (INTC_PSR752)	16	R/W	8000h	23.6.10/ 543
642	INTC Priority Select Register (INTC_PSR753)	16	R/W	8000h	23.6.10/ 543
644	INTC Priority Select Register (INTC_PSR754)	16	R/W	8000h	23.6.10/ 543
646	INTC Priority Select Register (INTC_PSR755)	16	R/W	8000h	23.6.10/ 543
648	INTC Priority Select Register (INTC_PSR756)	16	R/W	8000h	23.6.10/ 543
64A	INTC Priority Select Register (INTC_PSR757)	16	R/W	8000h	23.6.10/ 543
64C	INTC Priority Select Register (INTC_PSR758)	16	R/W	8000h	23.6.10/ 543
64E	INTC Priority Select Register (INTC_PSR759)	16	R/W	8000h	23.6.10/ 543
650	INTC Priority Select Register (INTC_PSR760)	16	R/W	8000h	23.6.10/ 543
652	INTC Priority Select Register (INTC_PSR761)	16	R/W	8000h	23.6.10/ 543
654	INTC Priority Select Register (INTC_PSR762)	16	R/W	8000h	23.6.10/ 543
656	INTC Priority Select Register (INTC_PSR763)	16	R/W	8000h	23.6.10/ 543
658	INTC Priority Select Register (INTC_PSR764)	16	R/W	8000h	23.6.10/ 543
65A	INTC Priority Select Register (INTC_PSR765)	16	R/W	8000h	23.6.10/ 543
65C	INTC Priority Select Register (INTC_PSR766)	16	R/W	8000h	23.6.10/ 543
65E	INTC Priority Select Register (INTC_PSR767)	16	R/W	8000h	23.6.10/ 543
660	INTC Priority Select Register (INTC_PSR768)	16	R/W	8000h	23.6.10/ 543
662	INTC Priority Select Register (INTC_PSR769)	16	R/W	8000h	23.6.10/ 543
664	INTC Priority Select Register (INTC_PSR770)	16	R/W	8000h	23.6.10/ 543
666	INTC Priority Select Register (INTC_PSR771)	16	R/W	8000h	23.6.10/ 543
668	INTC Priority Select Register (INTC_PSR772)	16	R/W	8000h	23.6.10/ 543
66A	INTC Priority Select Register (INTC_PSR773)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
66C	INTC Priority Select Register (INTC_PSR774)	16	R/W	8000h	23.6.10/ 543
66E	INTC Priority Select Register (INTC_PSR775)	16	R/W	8000h	23.6.10/ 543
670	INTC Priority Select Register (INTC_PSR776)	16	R/W	8000h	23.6.10/ 543
672	INTC Priority Select Register (INTC_PSR777)	16	R/W	8000h	23.6.10/ 543
674	INTC Priority Select Register (INTC_PSR778)	16	R/W	8000h	23.6.10/ 543
676	INTC Priority Select Register (INTC_PSR779)	16	R/W	8000h	23.6.10/ 543
678	INTC Priority Select Register (INTC_PSR780)	16	R/W	8000h	23.6.10/ 543
67A	INTC Priority Select Register (INTC_PSR781)	16	R/W	8000h	23.6.10/ 543
67C	INTC Priority Select Register (INTC_PSR782)	16	R/W	8000h	23.6.10/ 543
67E	INTC Priority Select Register (INTC_PSR783)	16	R/W	8000h	23.6.10/ 543
680	INTC Priority Select Register (INTC_PSR784)	16	R/W	8000h	23.6.10/ 543
682	INTC Priority Select Register (INTC_PSR785)	16	R/W	8000h	23.6.10/ 543
684	INTC Priority Select Register (INTC_PSR786)	16	R/W	8000h	23.6.10/ 543
686	INTC Priority Select Register (INTC_PSR787)	16	R/W	8000h	23.6.10/ 543
688	INTC Priority Select Register (INTC_PSR788)	16	R/W	8000h	23.6.10/ 543
68A	INTC Priority Select Register (INTC_PSR789)	16	R/W	8000h	23.6.10/ 543
68C	INTC Priority Select Register (INTC_PSR790)	16	R/W	8000h	23.6.10/ 543
68E	INTC Priority Select Register (INTC_PSR791)	16	R/W	8000h	23.6.10/ 543
690	INTC Priority Select Register (INTC_PSR792)	16	R/W	8000h	23.6.10/ 543
692	INTC Priority Select Register (INTC_PSR793)	16	R/W	8000h	23.6.10/ 543
694	INTC Priority Select Register (INTC_PSR794)	16	R/W	8000h	23.6.10/ 543
696	INTC Priority Select Register (INTC_PSR795)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
698	INTC Priority Select Register (INTC_PSR796)	16	R/W	8000h	23.6.10/ 543
69A	INTC Priority Select Register (INTC_PSR797)	16	R/W	8000h	23.6.10/ 543
69C	INTC Priority Select Register (INTC_PSR798)	16	R/W	8000h	23.6.10/ 543
69E	INTC Priority Select Register (INTC_PSR799)	16	R/W	8000h	23.6.10/ 543
6A0	INTC Priority Select Register (INTC_PSR800)	16	R/W	8000h	23.6.10/ 543
6A2	INTC Priority Select Register (INTC_PSR801)	16	R/W	8000h	23.6.10/ 543
6A4	INTC Priority Select Register (INTC_PSR802)	16	R/W	8000h	23.6.10/ 543
6A6	INTC Priority Select Register (INTC_PSR803)	16	R/W	8000h	23.6.10/ 543
6A8	INTC Priority Select Register (INTC_PSR804)	16	R/W	8000h	23.6.10/ 543
6AA	INTC Priority Select Register (INTC_PSR805)	16	R/W	8000h	23.6.10/ 543
6AC	INTC Priority Select Register (INTC_PSR806)	16	R/W	8000h	23.6.10/ 543
6AE	INTC Priority Select Register (INTC_PSR807)	16	R/W	8000h	23.6.10/ 543
6B0	INTC Priority Select Register (INTC_PSR808)	16	R/W	8000h	23.6.10/ 543
6B2	INTC Priority Select Register (INTC_PSR809)	16	R/W	8000h	23.6.10/ 543
6B4	INTC Priority Select Register (INTC_PSR810)	16	R/W	8000h	23.6.10/ 543
6B6	INTC Priority Select Register (INTC_PSR811)	16	R/W	8000h	23.6.10/ 543
6B8	INTC Priority Select Register (INTC_PSR812)	16	R/W	8000h	23.6.10/ 543
6BA	INTC Priority Select Register (INTC_PSR813)	16	R/W	8000h	23.6.10/ 543
6BC	INTC Priority Select Register (INTC_PSR814)	16	R/W	8000h	23.6.10/ 543
6BE	INTC Priority Select Register (INTC_PSR815)	16	R/W	8000h	23.6.10/ 543
6C0	INTC Priority Select Register (INTC_PSR816)	16	R/W	8000h	23.6.10/ 543
6C2	INTC Priority Select Register (INTC_PSR817)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6C4	INTC Priority Select Register (INTC_PSR818)	16	R/W	8000h	23.6.10/ 543
6C6	INTC Priority Select Register (INTC_PSR819)	16	R/W	8000h	23.6.10/ 543
6C8	INTC Priority Select Register (INTC_PSR820)	16	R/W	8000h	23.6.10/ 543
6CA	INTC Priority Select Register (INTC_PSR821)	16	R/W	8000h	23.6.10/ 543
6CC	INTC Priority Select Register (INTC_PSR822)	16	R/W	8000h	23.6.10/ 543
6CE	INTC Priority Select Register (INTC_PSR823)	16	R/W	8000h	23.6.10/ 543
6D0	INTC Priority Select Register (INTC_PSR824)	16	R/W	8000h	23.6.10/ 543
6D2	INTC Priority Select Register (INTC_PSR825)	16	R/W	8000h	23.6.10/ 543
6D4	INTC Priority Select Register (INTC_PSR826)	16	R/W	8000h	23.6.10/ 543
6D6	INTC Priority Select Register (INTC_PSR827)	16	R/W	8000h	23.6.10/ 543
6D8	INTC Priority Select Register (INTC_PSR828)	16	R/W	8000h	23.6.10/ 543
6DA	INTC Priority Select Register (INTC_PSR829)	16	R/W	8000h	23.6.10/ 543
6DC	INTC Priority Select Register (INTC_PSR830)	16	R/W	8000h	23.6.10/ 543
6DE	INTC Priority Select Register (INTC_PSR831)	16	R/W	8000h	23.6.10/ 543
6E0	INTC Priority Select Register (INTC_PSR832)	16	R/W	8000h	23.6.10/ 543
6E2	INTC Priority Select Register (INTC_PSR833)	16	R/W	8000h	23.6.10/ 543
6E4	INTC Priority Select Register (INTC_PSR834)	16	R/W	8000h	23.6.10/ 543
6E6	INTC Priority Select Register (INTC_PSR835)	16	R/W	8000h	23.6.10/ 543
6E8	INTC Priority Select Register (INTC_PSR836)	16	R/W	8000h	23.6.10/ 543
6EA	INTC Priority Select Register (INTC_PSR837)	16	R/W	8000h	23.6.10/ 543
6EC	INTC Priority Select Register (INTC_PSR838)	16	R/W	8000h	23.6.10/ 543
6EE	INTC Priority Select Register (INTC_PSR839)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6F0	INTC Priority Select Register (INTC_PSR840)	16	R/W	8000h	23.6.10/ 543
6F2	INTC Priority Select Register (INTC_PSR841)	16	R/W	8000h	23.6.10/ 543
6F4	INTC Priority Select Register (INTC_PSR842)	16	R/W	8000h	23.6.10/ 543
6F6	INTC Priority Select Register (INTC_PSR843)	16	R/W	8000h	23.6.10/ 543
6F8	INTC Priority Select Register (INTC_PSR844)	16	R/W	8000h	23.6.10/ 543
6FA	INTC Priority Select Register (INTC_PSR845)	16	R/W	8000h	23.6.10/ 543
6FC	INTC Priority Select Register (INTC_PSR846)	16	R/W	8000h	23.6.10/ 543
6FE	INTC Priority Select Register (INTC_PSR847)	16	R/W	8000h	23.6.10/ 543
700	INTC Priority Select Register (INTC_PSR848)	16	R/W	8000h	23.6.10/ 543
702	INTC Priority Select Register (INTC_PSR849)	16	R/W	8000h	23.6.10/ 543
704	INTC Priority Select Register (INTC_PSR850)	16	R/W	8000h	23.6.10/ 543
706	INTC Priority Select Register (INTC_PSR851)	16	R/W	8000h	23.6.10/ 543
708	INTC Priority Select Register (INTC_PSR852)	16	R/W	8000h	23.6.10/ 543
70A	INTC Priority Select Register (INTC_PSR853)	16	R/W	8000h	23.6.10/ 543
70C	INTC Priority Select Register (INTC_PSR854)	16	R/W	8000h	23.6.10/ 543
70E	INTC Priority Select Register (INTC_PSR855)	16	R/W	8000h	23.6.10/ 543
710	INTC Priority Select Register (INTC_PSR856)	16	R/W	8000h	23.6.10/ 543
712	INTC Priority Select Register (INTC_PSR857)	16	R/W	8000h	23.6.10/ 543
714	INTC Priority Select Register (INTC_PSR858)	16	R/W	8000h	23.6.10/ 543
716	INTC Priority Select Register (INTC_PSR859)	16	R/W	8000h	23.6.10/ 543
718	INTC Priority Select Register (INTC_PSR860)	16	R/W	8000h	23.6.10/ 543
71A	INTC Priority Select Register (INTC_PSR861)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
71C	INTC Priority Select Register (INTC_PSR862)	16	R/W	8000h	23.6.10/ 543
71E	INTC Priority Select Register (INTC_PSR863)	16	R/W	8000h	23.6.10/ 543
720	INTC Priority Select Register (INTC_PSR864)	16	R/W	8000h	23.6.10/ 543
722	INTC Priority Select Register (INTC_PSR865)	16	R/W	8000h	23.6.10/ 543
724	INTC Priority Select Register (INTC_PSR866)	16	R/W	8000h	23.6.10/ 543
726	INTC Priority Select Register (INTC_PSR867)	16	R/W	8000h	23.6.10/ 543
728	INTC Priority Select Register (INTC_PSR868)	16	R/W	8000h	23.6.10/ 543
72A	INTC Priority Select Register (INTC_PSR869)	16	R/W	8000h	23.6.10/ 543
72C	INTC Priority Select Register (INTC_PSR870)	16	R/W	8000h	23.6.10/ 543
72E	INTC Priority Select Register (INTC_PSR871)	16	R/W	8000h	23.6.10/ 543
730	INTC Priority Select Register (INTC_PSR872)	16	R/W	8000h	23.6.10/ 543
732	INTC Priority Select Register (INTC_PSR873)	16	R/W	8000h	23.6.10/ 543
734	INTC Priority Select Register (INTC_PSR874)	16	R/W	8000h	23.6.10/ 543
736	INTC Priority Select Register (INTC_PSR875)	16	R/W	8000h	23.6.10/ 543
738	INTC Priority Select Register (INTC_PSR876)	16	R/W	8000h	23.6.10/ 543
73A	INTC Priority Select Register (INTC_PSR877)	16	R/W	8000h	23.6.10/ 543
73C	INTC Priority Select Register (INTC_PSR878)	16	R/W	8000h	23.6.10/ 543
73E	INTC Priority Select Register (INTC_PSR879)	16	R/W	8000h	23.6.10/ 543
740	INTC Priority Select Register (INTC_PSR880)	16	R/W	8000h	23.6.10/ 543
742	INTC Priority Select Register (INTC_PSR881)	16	R/W	8000h	23.6.10/ 543
744	INTC Priority Select Register (INTC_PSR882)	16	R/W	8000h	23.6.10/ 543
746	INTC Priority Select Register (INTC_PSR883)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
748	INTC Priority Select Register (INTC_PSR884)	16	R/W	8000h	23.6.10/ 543
74A	INTC Priority Select Register (INTC_PSR885)	16	R/W	8000h	23.6.10/ 543
74C	INTC Priority Select Register (INTC_PSR886)	16	R/W	8000h	23.6.10/ 543
74E	INTC Priority Select Register (INTC_PSR887)	16	R/W	8000h	23.6.10/ 543
750	INTC Priority Select Register (INTC_PSR888)	16	R/W	8000h	23.6.10/ 543
752	INTC Priority Select Register (INTC_PSR889)	16	R/W	8000h	23.6.10/ 543
754	INTC Priority Select Register (INTC_PSR890)	16	R/W	8000h	23.6.10/ 543
756	INTC Priority Select Register (INTC_PSR891)	16	R/W	8000h	23.6.10/ 543
758	INTC Priority Select Register (INTC_PSR892)	16	R/W	8000h	23.6.10/ 543
75A	INTC Priority Select Register (INTC_PSR893)	16	R/W	8000h	23.6.10/ 543
75C	INTC Priority Select Register (INTC_PSR894)	16	R/W	8000h	23.6.10/ 543
75E	INTC Priority Select Register (INTC_PSR895)	16	R/W	8000h	23.6.10/ 543
760	INTC Priority Select Register (INTC_PSR896)	16	R/W	8000h	23.6.10/ 543
762	INTC Priority Select Register (INTC_PSR897)	16	R/W	8000h	23.6.10/ 543
764	INTC Priority Select Register (INTC_PSR898)	16	R/W	8000h	23.6.10/ 543
766	INTC Priority Select Register (INTC_PSR899)	16	R/W	8000h	23.6.10/ 543
768	INTC Priority Select Register (INTC_PSR900)	16	R/W	8000h	23.6.10/ 543
76A	INTC Priority Select Register (INTC_PSR901)	16	R/W	8000h	23.6.10/ 543
76C	INTC Priority Select Register (INTC_PSR902)	16	R/W	8000h	23.6.10/ 543
76E	INTC Priority Select Register (INTC_PSR903)	16	R/W	8000h	23.6.10/ 543
770	INTC Priority Select Register (INTC_PSR904)	16	R/W	8000h	23.6.10/ 543
772	INTC Priority Select Register (INTC_PSR905)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
774	INTC Priority Select Register (INTC_PSR906)	16	R/W	8000h	23.6.10/ 543
776	INTC Priority Select Register (INTC_PSR907)	16	R/W	8000h	23.6.10/ 543
778	INTC Priority Select Register (INTC_PSR908)	16	R/W	8000h	23.6.10/ 543
77A	INTC Priority Select Register (INTC_PSR909)	16	R/W	8000h	23.6.10/ 543
77C	INTC Priority Select Register (INTC_PSR910)	16	R/W	8000h	23.6.10/ 543
77E	INTC Priority Select Register (INTC_PSR911)	16	R/W	8000h	23.6.10/ 543
780	INTC Priority Select Register (INTC_PSR912)	16	R/W	8000h	23.6.10/ 543
782	INTC Priority Select Register (INTC_PSR913)	16	R/W	8000h	23.6.10/ 543
784	INTC Priority Select Register (INTC_PSR914)	16	R/W	8000h	23.6.10/ 543
786	INTC Priority Select Register (INTC_PSR915)	16	R/W	8000h	23.6.10/ 543
788	INTC Priority Select Register (INTC_PSR916)	16	R/W	8000h	23.6.10/ 543
78A	INTC Priority Select Register (INTC_PSR917)	16	R/W	8000h	23.6.10/ 543
78C	INTC Priority Select Register (INTC_PSR918)	16	R/W	8000h	23.6.10/ 543
78E	INTC Priority Select Register (INTC_PSR919)	16	R/W	8000h	23.6.10/ 543
790	INTC Priority Select Register (INTC_PSR920)	16	R/W	8000h	23.6.10/ 543
792	INTC Priority Select Register (INTC_PSR921)	16	R/W	8000h	23.6.10/ 543
794	INTC Priority Select Register (INTC_PSR922)	16	R/W	8000h	23.6.10/ 543
796	INTC Priority Select Register (INTC_PSR923)	16	R/W	8000h	23.6.10/ 543
798	INTC Priority Select Register (INTC_PSR924)	16	R/W	8000h	23.6.10/ 543
79A	INTC Priority Select Register (INTC_PSR925)	16	R/W	8000h	23.6.10/ 543
79C	INTC Priority Select Register (INTC_PSR926)	16	R/W	8000h	23.6.10/ 543
79E	INTC Priority Select Register (INTC_PSR927)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7A0	INTC Priority Select Register (INTC_PSR928)	16	R/W	8000h	23.6.10/ 543
7A2	INTC Priority Select Register (INTC_PSR929)	16	R/W	8000h	23.6.10/ 543
7A4	INTC Priority Select Register (INTC_PSR930)	16	R/W	8000h	23.6.10/ 543
7A6	INTC Priority Select Register (INTC_PSR931)	16	R/W	8000h	23.6.10/ 543
7A8	INTC Priority Select Register (INTC_PSR932)	16	R/W	8000h	23.6.10/ 543
7AA	INTC Priority Select Register (INTC_PSR933)	16	R/W	8000h	23.6.10/ 543
7AC	INTC Priority Select Register (INTC_PSR934)	16	R/W	8000h	23.6.10/ 543
7AE	INTC Priority Select Register (INTC_PSR935)	16	R/W	8000h	23.6.10/ 543
7B0	INTC Priority Select Register (INTC_PSR936)	16	R/W	8000h	23.6.10/ 543
7B2	INTC Priority Select Register (INTC_PSR937)	16	R/W	8000h	23.6.10/ 543
7B4	INTC Priority Select Register (INTC_PSR938)	16	R/W	8000h	23.6.10/ 543
7B6	INTC Priority Select Register (INTC_PSR939)	16	R/W	8000h	23.6.10/ 543
7B8	INTC Priority Select Register (INTC_PSR940)	16	R/W	8000h	23.6.10/ 543
7BA	INTC Priority Select Register (INTC_PSR941)	16	R/W	8000h	23.6.10/ 543
7BC	INTC Priority Select Register (INTC_PSR942)	16	R/W	8000h	23.6.10/ 543
7BE	INTC Priority Select Register (INTC_PSR943)	16	R/W	8000h	23.6.10/ 543
7C0	INTC Priority Select Register (INTC_PSR944)	16	R/W	8000h	23.6.10/ 543
7C2	INTC Priority Select Register (INTC_PSR945)	16	R/W	8000h	23.6.10/ 543
7C4	INTC Priority Select Register (INTC_PSR946)	16	R/W	8000h	23.6.10/ 543
7C6	INTC Priority Select Register (INTC_PSR947)	16	R/W	8000h	23.6.10/ 543
7C8	INTC Priority Select Register (INTC_PSR948)	16	R/W	8000h	23.6.10/ 543
7CA	INTC Priority Select Register (INTC_PSR949)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7CC	INTC Priority Select Register (INTC_PSR950)	16	R/W	8000h	23.6.10/ 543
7CE	INTC Priority Select Register (INTC_PSR951)	16	R/W	8000h	23.6.10/ 543
7D0	INTC Priority Select Register (INTC_PSR952)	16	R/W	8000h	23.6.10/ 543
7D2	INTC Priority Select Register (INTC_PSR953)	16	R/W	8000h	23.6.10/ 543
7D4	INTC Priority Select Register (INTC_PSR954)	16	R/W	8000h	23.6.10/ 543
7D6	INTC Priority Select Register (INTC_PSR955)	16	R/W	8000h	23.6.10/ 543
7D8	INTC Priority Select Register (INTC_PSR956)	16	R/W	8000h	23.6.10/ 543
7DA	INTC Priority Select Register (INTC_PSR957)	16	R/W	8000h	23.6.10/ 543
7DC	INTC Priority Select Register (INTC_PSR958)	16	R/W	8000h	23.6.10/ 543
7DE	INTC Priority Select Register (INTC_PSR959)	16	R/W	8000h	23.6.10/ 543
7E0	INTC Priority Select Register (INTC_PSR960)	16	R/W	8000h	23.6.10/ 543
7E2	INTC Priority Select Register (INTC_PSR961)	16	R/W	8000h	23.6.10/ 543
7E4	INTC Priority Select Register (INTC_PSR962)	16	R/W	8000h	23.6.10/ 543
7E6	INTC Priority Select Register (INTC_PSR963)	16	R/W	8000h	23.6.10/ 543
7E8	INTC Priority Select Register (INTC_PSR964)	16	R/W	8000h	23.6.10/ 543
7EA	INTC Priority Select Register (INTC_PSR965)	16	R/W	8000h	23.6.10/ 543
7EC	INTC Priority Select Register (INTC_PSR966)	16	R/W	8000h	23.6.10/ 543
7EE	INTC Priority Select Register (INTC_PSR967)	16	R/W	8000h	23.6.10/ 543
7F0	INTC Priority Select Register (INTC_PSR968)	16	R/W	8000h	23.6.10/ 543
7F2	INTC Priority Select Register (INTC_PSR969)	16	R/W	8000h	23.6.10/ 543
7F4	INTC Priority Select Register (INTC_PSR970)	16	R/W	8000h	23.6.10/ 543
7F6	INTC Priority Select Register (INTC_PSR971)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
7F8	INTC Priority Select Register (INTC_PSR972)	16	R/W	8000h	23.6.10/ 543
7FA	INTC Priority Select Register (INTC_PSR973)	16	R/W	8000h	23.6.10/ 543
7FC	INTC Priority Select Register (INTC_PSR974)	16	R/W	8000h	23.6.10/ 543
7FE	INTC Priority Select Register (INTC_PSR975)	16	R/W	8000h	23.6.10/ 543
800	INTC Priority Select Register (INTC_PSR976)	16	R/W	8000h	23.6.10/ 543
802	INTC Priority Select Register (INTC_PSR977)	16	R/W	8000h	23.6.10/ 543
804	INTC Priority Select Register (INTC_PSR978)	16	R/W	8000h	23.6.10/ 543
806	INTC Priority Select Register (INTC_PSR979)	16	R/W	8000h	23.6.10/ 543
808	INTC Priority Select Register (INTC_PSR980)	16	R/W	8000h	23.6.10/ 543
80A	INTC Priority Select Register (INTC_PSR981)	16	R/W	8000h	23.6.10/ 543
80C	INTC Priority Select Register (INTC_PSR982)	16	R/W	8000h	23.6.10/ 543
80E	INTC Priority Select Register (INTC_PSR983)	16	R/W	8000h	23.6.10/ 543
810	INTC Priority Select Register (INTC_PSR984)	16	R/W	8000h	23.6.10/ 543
812	INTC Priority Select Register (INTC_PSR985)	16	R/W	8000h	23.6.10/ 543
814	INTC Priority Select Register (INTC_PSR986)	16	R/W	8000h	23.6.10/ 543
816	INTC Priority Select Register (INTC_PSR987)	16	R/W	8000h	23.6.10/ 543
818	INTC Priority Select Register (INTC_PSR988)	16	R/W	8000h	23.6.10/ 543
81A	INTC Priority Select Register (INTC_PSR989)	16	R/W	8000h	23.6.10/ 543
81C	INTC Priority Select Register (INTC_PSR990)	16	R/W	8000h	23.6.10/ 543
81E	INTC Priority Select Register (INTC_PSR991)	16	R/W	8000h	23.6.10/ 543
820	INTC Priority Select Register (INTC_PSR992)	16	R/W	8000h	23.6.10/ 543
822	INTC Priority Select Register (INTC_PSR993)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
824	INTC Priority Select Register (INTC_PSR994)	16	R/W	8000h	23.6.10/ 543
826	INTC Priority Select Register (INTC_PSR995)	16	R/W	8000h	23.6.10/ 543
828	INTC Priority Select Register (INTC_PSR996)	16	R/W	8000h	23.6.10/ 543
82A	INTC Priority Select Register (INTC_PSR997)	16	R/W	8000h	23.6.10/ 543
82C	INTC Priority Select Register (INTC_PSR998)	16	R/W	8000h	23.6.10/ 543
82E	INTC Priority Select Register (INTC_PSR999)	16	R/W	8000h	23.6.10/ 543
830	INTC Priority Select Register (INTC_PSR1000)	16	R/W	8000h	23.6.10/ 543
832	INTC Priority Select Register (INTC_PSR1001)	16	R/W	8000h	23.6.10/ 543
834	INTC Priority Select Register (INTC_PSR1002)	16	R/W	8000h	23.6.10/ 543
836	INTC Priority Select Register (INTC_PSR1003)	16	R/W	8000h	23.6.10/ 543
838	INTC Priority Select Register (INTC_PSR1004)	16	R/W	8000h	23.6.10/ 543
83A	INTC Priority Select Register (INTC_PSR1005)	16	R/W	8000h	23.6.10/ 543
83C	INTC Priority Select Register (INTC_PSR1006)	16	R/W	8000h	23.6.10/ 543
83E	INTC Priority Select Register (INTC_PSR1007)	16	R/W	8000h	23.6.10/ 543
840	INTC Priority Select Register (INTC_PSR1008)	16	R/W	8000h	23.6.10/ 543
842	INTC Priority Select Register (INTC_PSR1009)	16	R/W	8000h	23.6.10/ 543
844	INTC Priority Select Register (INTC_PSR1010)	16	R/W	8000h	23.6.10/ 543
846	INTC Priority Select Register (INTC_PSR1011)	16	R/W	8000h	23.6.10/ 543
848	INTC Priority Select Register (INTC_PSR1012)	16	R/W	8000h	23.6.10/ 543
84A	INTC Priority Select Register (INTC_PSR1013)	16	R/W	8000h	23.6.10/ 543
84C	INTC Priority Select Register (INTC_PSR1014)	16	R/W	8000h	23.6.10/ 543
84E	INTC Priority Select Register (INTC_PSR1015)	16	R/W	8000h	23.6.10/ 543

Table continues on the next page...

INTC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
850	INTC Priority Select Register (INTC_PSR1016)	16	R/W	8000h	23.6.10/ 543
852	INTC Priority Select Register (INTC_PSR1017)	16	R/W	8000h	23.6.10/ 543
854	INTC Priority Select Register (INTC_PSR1018)	16	R/W	8000h	23.6.10/ 543
856	INTC Priority Select Register (INTC_PSR1019)	16	R/W	8000h	23.6.10/ 543
858	INTC Priority Select Register (INTC_PSR1020)	16	R/W	8000h	23.6.10/ 543
85A	INTC Priority Select Register (INTC_PSR1021)	16	R/W	8000h	23.6.10/ 543
85C	INTC Priority Select Register (INTC_PSR1022)	16	R/W	8000h	23.6.10/ 543
85E	INTC Priority Select Register (INTC_PSR1023)	16	R/W	8000h	23.6.10/ 543

23.6.1 INTC Block Configuration Register (INTC_BCR)

The Block Configuration Register is used to configure options of the INTC.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] has write access, otherwise, a termination error is asserted.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			0	0			0	0				0			
W												HVEN1				HVEN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

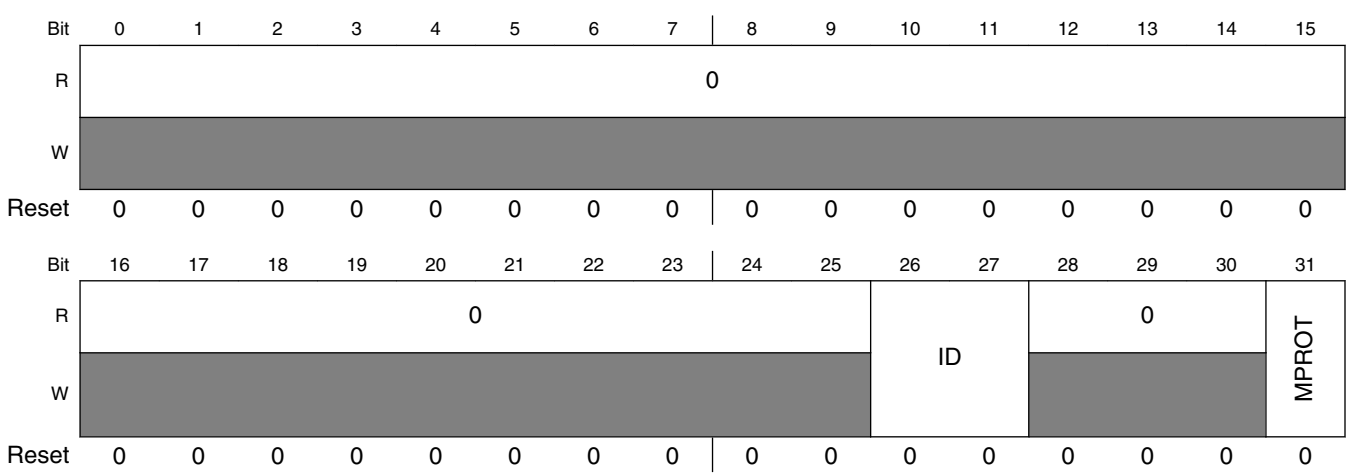
INTC_BCR field descriptions

Field	Description
0–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 HVEN1	Hardware vector enable for processor 1. Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HVEN0	Hardware vector enable for processor 0. Controls whether the INTC is in hardware vector mode or software vector mode. See Modes of operation for the details of the handshaking with the processor in each mode. 0 Software vector mode 1 Hardware vector mode

23.6.2 INTC Master Protection Register (INTC_MPROT)

The Master Protection Register is used to protect cores from writing to other cores INTC registers.

Address: 0h base + 4h offset = 4h



INTC_MPROT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 ID	ID This register can only be updated if MPROT=0 or if processor master ID=ID. When MPROT is set, the ID should also be updated to indicate which master can write to any register in the INTC memory map. Only the field values related to core bus masters can be selected. Non-core bus masters cannot be selected and would result in a bus access error. 00 master0 (reset default) 01 master1 10 master2 11 master3
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MPROT	Master Protection Controls what programming space a given master can write to. If MPROT is disabled, then all masters can write to any valid entry of the INTC memory map. If MPROT is enabled, INTC_MPROT[ID] can write to any valid entry of the INTC memory map but other masters can only write to their assigned resources. Once MPROT is enabled by a master, only that master may modify INTC_MPROT. 0 Master programming space protection disabled 1 Master programming space protection enabled

23.6.3 INTC Current Priority Register for Processor 0 (INTC_CPR0)

The INTC_CPRn masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*n*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*n*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*n*) is written, the LIFO is popped into the INTC_CPR*n* PRI field. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

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

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See Ensuring coherency for example code to ensure coherency.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												PRI			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

INTC_CPR0 field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 PRI	Priority of the currently executing ISR, according to the field values 15 (highest priority) down to 0 (lowest priority).

23.6.4 INTC Current Priority Register for Processor 1 (INTC_CPR1)

The INTC_CPRn masks any peripheral or software-settable interrupt request that is set at the same or lower priority as the current value of the INTC_CPR*n*[PRI] field from generating an interrupt request to the processor. When the INTC Interrupt Acknowledge register (INTC_IACKR*n*) is read in software vector mode or the interrupt acknowledge signal from the processor is asserted in hardware vector mode, the value of PRI is pushed onto the LIFO, and PRI is updated with the priority of the preempting interrupt request. When the INTC end-of-interrupt register (INTC_EOIR*n*) is written, the LIFO is popped into the INTC_CPR*n* PRI field. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

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

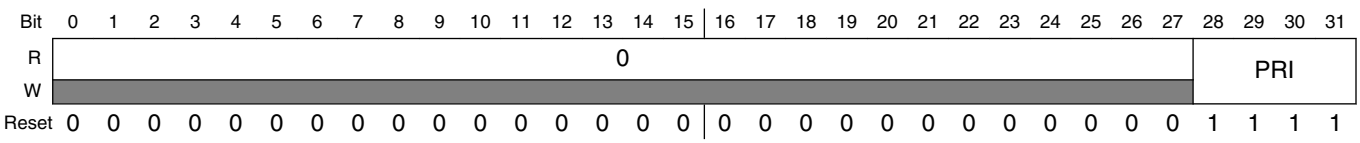
If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

NOTE

A store to modify the PRI field that closely precedes or follows an access to a shared resource may result in a non-coherent access to that resource. See Ensuring coherency for example code to ensure coherency.

Address: 0h base + 14h offset = 14h



INTC_CPR1 field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 PRI	Priority of the currently executing ISR, according to the field values 15 (highest priority) down to 0 (lowest priority).

23.6.5 INTC Interrupt Acknowledge Register for Processor 0 (INTC_IACKR0)

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR*n* has side effects from reads unless INTC_MPROT[MPROT]=1 and processor with master ID (0-3) != INTC_MPROT[ID]. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKRn does not have side effects in hardware vector mode.

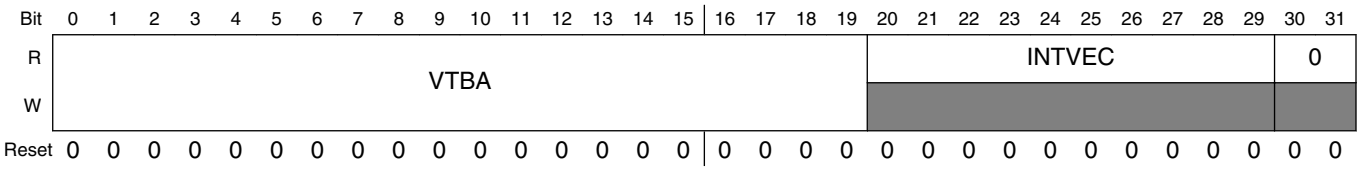
If INTC_MPROT[MPROT]=0 (disabled), all processors have read and write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has read and write access, otherwise, a termination error is asserted.

NOTE

When the corresponding INTC_BCR[HVENn] = 1, a read of the INTC_IACKR has no side effects.

Address: 0h base + 20h offset = 20h



INTC_IACKR0 field descriptions

Field	Description
0–19 VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
20–29 INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

23.6.6 INTC Interrupt Acknowledge Register for Processor 1 (INTC_IACKR1)

The Interrupt Acknowledge Register for Processor *n* provides a value which can be used to load the address of an ISR from a vector table. The vector table can be composed of addresses of the ISRs specific to their respective interrupt vectors.

Also, in software vector mode, the INTC_IACKR*n* has side effects from reads unless INTC_MPROT[MPROT]=1 and processor with master ID (0-3) != INTC_MPROT[ID]. Therefore, it must not be read speculatively while in this mode. The side effects are the same regardless of the size of the read. Reading the INTC_IACKRn does not have side effects in hardware vector mode.

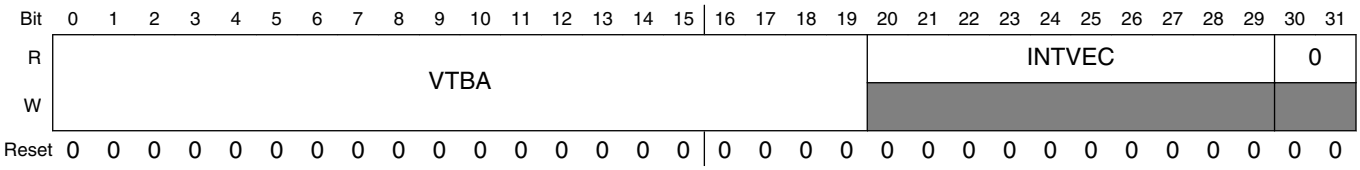
If INTC_MPROT[MPROT]=0 (disabled), all processors have read and write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has read and write access, otherwise, a termination error is asserted.

NOTE

When the corresponding INTC_BCR[HVENn] = 1, a read of the INTC_IACKR has no side effects.

Address: 0h base + 24h offset = 24h



INTC_IACKR1 field descriptions

Field	Description
0–19 VTBA	Vector table base address. Can be the base address of a vector table of addresses of ISRs.
20–29 INTVEC	Interrupt vector. Vector of the peripheral or software-settable interrupt request that caused the interrupt request to the processor. When the interrupt request to the processor asserts, the INTVEC is updated, whether the INTC is in software or hardware vector mode.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

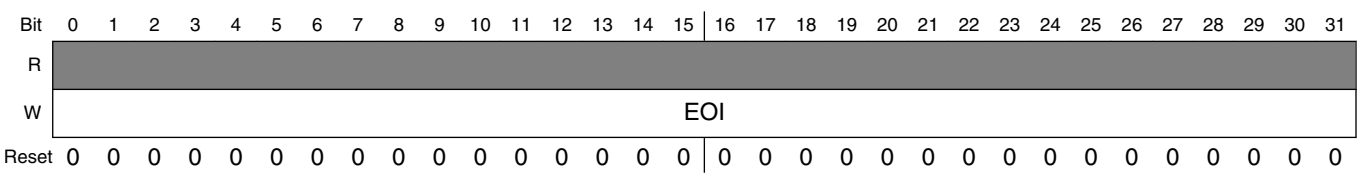
23.6.7 INTC End Of Interrupt Register for Processor 0 (INTC_EOIR0)

Writing to the INTC_EOIR*n* signals the end of the servicing of the interrupt request. When the INTC_EOIR*n* is written, the priority last pushed on the LIFO is popped into INTC_CPR*n*. The values and size of data written to the INTC_EOIR*n* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*n* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

Address: 0h base + 30h offset = 30h



INTC_EOIR0 field descriptions

Field	Description
0–31 EOI	End of Interrupt. Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

23.6.8 INTC End Of Interrupt Register for Processor 1 (INTC_EOIR1)

Writing to the INTC_EOIR*n* signals the end of the servicing of the interrupt request. When the INTC_EOIR*n* is written, the priority last pushed on the LIFO is popped into INTC_CPR*n*. The values and size of data written to the INTC_EOIR*n* are ignored. Those values and sizes written to this register neither update the INTC_EOIR*n* contents nor affect whether the LIFO pops. Typically, when you write to this register, write four all-zero bytes.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access to this register.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access, otherwise, a termination error is asserted.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	EOI																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

INTC_EOIR1 field descriptions

Field	Description
0–31 EOI	End of Interrupt. Write four all-zero bytes to this field to signal the end of the servicing of an interrupt request.

23.6.9 INTC Software Set/Clear Interrupt Register (INTC_SSCIR*n*)

The INTC_SSCIR registers support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. Unlike the SWT bit in the PSRs, the INTC_SSCIR registers do not require a read-modify-write. With the exception of being set by software, this flag bit behaves the same as a flag bit set within a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a 1 to SET will leave SET unchanged at 0 but will set CLR. Writing a 0 to SET will have no effect. CLR is the flag bit. Writing a 1 to CLR will clear it. Writing a 0 to CLR will have no effect. If a 1 is written to a pair of SET and CLR bits at the same time, CLR is asserted, regardless of whether CLR was asserted before the write.

See Figure 23-2 for limitations on writing to this register.

Any processor will be allowed to write to any of the SET bits in the INTC_SSCIRn registers.

If INTC_MPROT[MPROT]=0 (disabled), all processors can write to the CLR bit.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor can write to the CLR bit, otherwise, a termination error is asserted.

NOTE

If INTC_MPROT[MPROT] is enabled, processors with master ID != INTC_MPROT[ID] must only write the INTC_SSCIRn registers using a byte operation. A write of any other size will result in a termination error.

Address: 0h base + 40h offset + (1d × i), where i=0d to 23d

Bit	0	1	2	3	4	5	6	7
Read	0						0	CLR
Write							SET	w1c
Reset	0	0	0	0	0	0	0	0

INTC_SSCIR*n* field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 SET	Set flag bits. Writing a 1 will set the corresponding CLR bit. Writing a 0 will have no effect. Each SET is read as a 0.
7 CLR	Clear flag bits. CLR is the flag bit. Writing a 1 to CLR will clear it provided that a 1 is not written simultaneously to its corresponding SET bit. Writing a 0 to CLR will have no effect. 0 Interrupt request not pending within INTC. 1 Interrupt request pending within INTC.

23.6.10 INTC Priority Select Register (INTC_PSR*n*)

The Priority Select Registers support the selection of an individual priority for each source of interrupt request, and the routing of the request to one or more processors. The unique vector of each peripheral or software-settable interrupt request determines which INTC_PSRn is assigned to that interrupt request. The software-settable interrupt requests are assigned the lowest numbered vectors, and their priorities are configured in the lowest numbered INTC_PSR registers. The peripheral interrupt requests are assigned to vectors immediately following the software-settable interrupt requests, and their priorities are configured in the immediately following INTC_PSR registers. The peripheral interrupt requests are assigned vectors 32-1023, and their priorities are configured in INTC_PSR32-1023, respectively.

If INTC_MPROT[MPROT]=0 (disabled), all processors have write access.

If INTC_MPROT[MPROT]=1 (enabled), the processor with master ID (0-3) = INTC_MPROT[ID] or the corresponding processor has write access to INTC_PSRn[PRC_SEL,SWT], otherwise, a termination error is asserted. When writing the INTC_PSRn[PRI], only the processor with master ID (0-3) = INTC_MPROT[ID] and the processor whose master ID (0-3) matches the setting of the INTC_PSRn[PRC_SEL] will be allowed to write the INTC_PSRn[PRI]. An attempt by any other processor to write the INTC_PSRn[PRI] will result in a termination error.

See Figure 23-2 for limitations on writing to this register.

NOTE

Unlike the peripheral interrupt request PSRs, there is no SWT bit in the PSRs corresponding to the software-settable interrupt requests.

NOTE

Interrupts are not detected while the PSRs are being written.

NOTE

If INTC_MPROT[MPROT] is enabled, the core with master ID (0-3) = INTC_MPROT[ID] will be able to write the INTC_PSRn with either byte, half-word or word operations. All other master IDs are restricted to byte operations to the INTC_PSRn[PRI] bits (i.e the odd addresses). Any other sized write operation, or a byte operation to the INTC_PSRn[PRC_SEL] (i.e. the even byte) will result in a termination error.

Address: 0h base + 60h offset + (2d × i), where i=0d to 1023d

Bit	0	1	2	3	4	5	6	7
Read	PRC_SELN0	PRC_SELN1	0	0	0			SWTN
Write								
Reset	1	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0				PRIN			
Write								
Reset	0	0	0	0	0	0	0	0

INTC_PSR*n* field descriptions

Field	Description
0 PRC_SELN0	Processor select 0. If an interrupt source is enabled, this field selects whether the interrupt request is to be sent to processor 0. 0 Interrupt request not sent to processor 0 1 Interrupt request sent to processor 0
1 PRC_SELN1	Processor select 1. If an interrupt source is enabled, this field selects whether the interrupt request is to be sent to processor 1. 0 Interrupt request not sent to processor 1 1 Interrupt request sent to processor 1
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SWTN	This bit is only available in PSRs that do not correspond to the SSCIRs. The SWT bit supports triggering an interrupt by software rather than hardware. This is an alternative to setting a flag bit in a peripheral. This flag bit generates an interrupt request within the INTC just like a peripheral interrupt request. Writing a '1' to SWT*n* sets the bit. On the interrupt acknowledge cycle, the SWT*n* bit is cleared. The SWT function works both in hardware and software vector mode. It is only cleared by the enabled Processor(s) defined in the PRC_SEL*n* field of the PSR.
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 PRIN	Priority select. Selects the priority for the interrupt requests. PRI has values 15 (highest priority) down to 0 (lowest priority).

23.7 Functional description

The functional description involves the areas of interrupt request sources, priority management, and handshaking with the processor. In addition, spaces in the memory map have been reserved for other possible implementations of the INTC.

23.7.1 Interrupt request sources

The INTC has two types of interrupt requests, peripheral and software-settable. These interrupt requests can assert on any clock cycle.

The INTC has no spurious vector support. Therefore, if an asserted peripheral or software-settable interrupt request, whose PRI*n* value in INTC_PSR*n* is higher than the PRI value in INTC_CPR*n*, negates before the interrupt request to the processor for that peripheral or software-settable interrupt request is acknowledged, then the interrupt request to the processor still can assert (or will remain asserted) for that peripheral or software-settable interrupt request. In this case, the interrupt vector will correspond to that peripheral or software-settable interrupt request. Also, the PRI value in the associated INTC_CPR*n* is updated with the corresponding PRI*n* value in INTC_PSR*n*.

Furthermore, clearing the peripheral interrupt request's enable bit in the peripheral, or (alternatively) setting its mask bit, has the same consequences as clearing its flag bit. Setting its enable bit or clearing its mask bit while its flag bit is asserted has the same effect on the INTC as an interrupt event setting the flag bit.

23.7.1.1 Peripheral interrupt requests

An interrupt event in a peripheral's hardware sets a flag bit which resides in that peripheral. The interrupt request from the peripheral is driven by that flag bit.

The time from when the peripheral starts to drive its peripheral interrupt request to the INTC, to the time that the INTC starts to drive the interrupt request to the processor, is three clocks.

23.7.1.2 Software-settable interrupt requests

The software set/clear interrupt registers (INTC_SSCIR*n*) support the setting or clearing of software-settable interrupt requests. These registers contain independent sets of bits to set and clear a corresponding flag bit by software. An interrupt request is triggered by software by writing a 1 to a SET bit in INTC_SSCIR*n*. This write sets the corresponding CLR bit, which is a flag bit, resulting in the interrupt request. The interrupt request is cleared by writing a 1 to the CLR bit. Specific behavior includes the following:

- Writing a 1 to SET leaves SET unchanged at 0 but sets the flag bit (which is the CLR bit).
- Writing a 0 to SET has no effect.

- Writing a 1 to CLR clears the flag (CLR) bit.
- Writing a 0 to CLR has no effect.
- If a 1 is written to a pair of SET and CLR bits at the same time, the flag (CLR) is set, regardless of whether CLR was asserted before the write.

The time from the write to the SET bit, to the time that the INTC starts to drive the interrupt request to the processor, is four clocks.

Any Processor n will be allowed to write to any of the SET bits in the Software Set/Clear Interrupt registers. The CLR bit will only be writable by the Processor n which has been assigned to handle that interrupt request, as determined by the setting of the INTC_PSRn[PRC_SELn] bits.

23.7.1.3 Unique vector for each interrupt request source

Each peripheral and software-settable interrupt request is assigned a hardwired unique 10-bit vector. Software-settable interrupts 0–23 are assigned vectors 0–23, respectively. The peripheral interrupt requests are assigned vectors from 24 to a number as high as needed to cover all peripheral interrupt requests.

23.7.2 Priority management

The asserted interrupt requests are compared to each other based on their PRI*n* and PRC_SEL*n* values set in INTC_PSR*n*. The result of that comparison also is compared to PRI in the associated INTC_CPR*n*. The results of those comparisons are used to manage the priority of the ISR being executed by the associated processor. The associated LIFO also assists in managing that priority.

23.7.2.1 Current priority and preemption

The priority arbitrator, selector, encoder, and comparator logic shown in Figure 23-1 are used to compare the priority of the asserted interrupt requests to the current priority. If the priority of any asserted peripheral or software-settable interrupt request is higher than the current priority for a given processor, then the interrupt request to the processor is asserted. Also, a unique vector for the preempting peripheral or software-settable interrupt request is generated for the associated INTC_IACKR*n*, and if in hardware vector mode, for the interrupt vector provided to the processor.

23.7.2.1.1 Priority tree

The priority tree for each processor compares all the priorities of all of the asserted interrupt requests assigned to that processor, both peripheral and software-settable. The output of the priority tree is the highest of those priorities assigned to a given processor. Also, any interrupt requests which have this highest priority are output as asserted interrupt requests to the associated selector logic.

23.7.2.1.2 Selector

If only one interrupt request from the associated priority tree is asserted, then it is passed as asserted to the associated encoder logic. If multiple interrupt requests from the associated priority tree are asserted, then only the one with the lowest vector is passed as asserted to the associated encoder logic. The lower vector is chosen regardless of the time order of the assertions of the peripheral or software-settable interrupt requests.

23.7.2.1.3 Encoder

The encoder logic generates the unique 10-bit vector for the asserted interrupt request from the request selector subblock for the associated processor.

23.7.2.1.4 Comparator

The comparator logic compares the highest priority output from the associated priority arbitrator subblock with PRI in the associated INTC_CPR*n*. If the comparator detects that this highest priority is higher than the current priority, then it asserts the interrupt request to the associated processor. This interrupt request to the processor asserts whether this highest priority is raised above the value of PRI in the associated INTC_CPR*n*, or the PRI value in the associated INTC_CPR*n* is lowered below this highest priority. This highest priority then becomes the new priority which is written to PRI in the associated INTC_CPR*n* when the interrupt request to the processor is acknowledged. Interrupt requests whose PRI*n* in INTC_PSR*n* are zero will not cause a preemption because their PRI*n* will not be higher than PRI in the associated INTC_CPR*n*.

Another function of the comparator is to signal an update of the INTC_IACKR*n* with the vector number of the first interrupt that arrives that has a priority higher than the current priority. Once the vector number and priority are captured, they cannot be superseded by a higher priority interrupt until an update of the INTC_CPR*n* occurs. In software vector mode, higher priority interrupts can supersede the previously captured interrupt vector number and priority until the time a hardware or software interrupt acknowledge is processed.

23.7.2.2 Stack (LIFO)

The LIFO stores the preempted PRI values from the associated INTC_CPR*n*. Therefore, because these priorities are stacked within the INTC, if interrupts need to be enabled during the ISR, at the beginning of the interrupt exception handler the PRI value in the associated INTC_CPR*n* does not need to be loaded from the associated INTC_CPR*n* and stored onto the context stack. Likewise at the end of the interrupt exception handler, the priority does not need to be loaded from the context stack and stored into the associated INTC_CPR*n*.

The PRI value in the associated INTC_CPR*n* is pushed onto the LIFO when the associated INTC_IACKR*n* is read in software vector mode or the interrupt acknowledge signal from the associated processor is asserted in hardware vector mode. The priority is popped into PRI in the associated INTC_CPR*n* whenever the associated INTC_EOIR*n* is written. An exception case in hardware vector mode to this behavior is described in Hardware vector mode.

Although the INTC supports up to 16 priorities, an ISR executing with PRI in the INTC_CPR*n* equal to 15 will not be preempted. Therefore, the LIFO supports the stacking of 15 priorities. However, the LIFO is only 14 entries deep. An entry for a priority of 0 is not needed because of the ways that pushing onto a full LIFO and popping an empty LIFO are treated. If the LIFO is pushed 15 or more times than it is popped, the priorities first pushed are overwritten. A priority of 0 would be an overwritten priority. However, the LIFO will pop zeroes if it is popped more times than it is pushed. Therefore, although a priority of 0 was overwritten, it is regenerated with the popping of an empty LIFO.

23.7.3 Handshaking with processor

This section describes handshaking with the processor.

23.7.3.1 Software vector mode handshaking

This section describes the software vector mode handshaking.

23.7.3.1.1 Acknowledging interrupt request to processor

The software vector mode handshaking can be used with processors that only support an interrupt request to them, or processors that support both an interrupt request by itself as well as an interrupt request with an interrupt vector. The software vector mode handshaking even supports processors that always expect an interrupt vector with the interrupt request to them. Refer to Figure 23-3.

A timing diagram of the interrupt request and acknowledge handshaking in software vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 23-3. The INTC examines the peripheral and software-settable interrupt requests. When it finds an asserted peripheral or software-settable interrupt request with a higher priority than PRI in the associated INTC_CPR*n*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*n* is updated with the preempting interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the processor is asserted. The rest of the handshaking is described in Software vector mode.

23.7.3.1.2 End of interrupt exception handler

Before the interrupt exception handling completes, INTC_EOIR*n* must be written. When it is written, the associated LIFO is popped so that the preempted priority is restored into PRI of the associated INTC_CPR*n*. Before it is written, the peripheral or software-settable flag bit must be cleared so that the peripheral or software-settable interrupt request is negated.

When returning from the preemption, the INTC does not search for the peripheral or software-settable interrupt request whose ISR was preempted. Depending on how much the ISR has progressed, that interrupt request may no longer even be asserted. When PRI in the associated INTC_CPR*n* is lowered to the priority of the preempted ISR, the interrupt request for the preempted ISR or any other asserted peripheral or software-settable interrupt request at or below that priority will not cause a preemption. Instead, after the restoration of the preempted context, the processor will return to the instruction address that it had been going to execute next, before it was preempted. This next instruction is part of the preempted ISR or the interrupt exception handler's prolog or epilog.

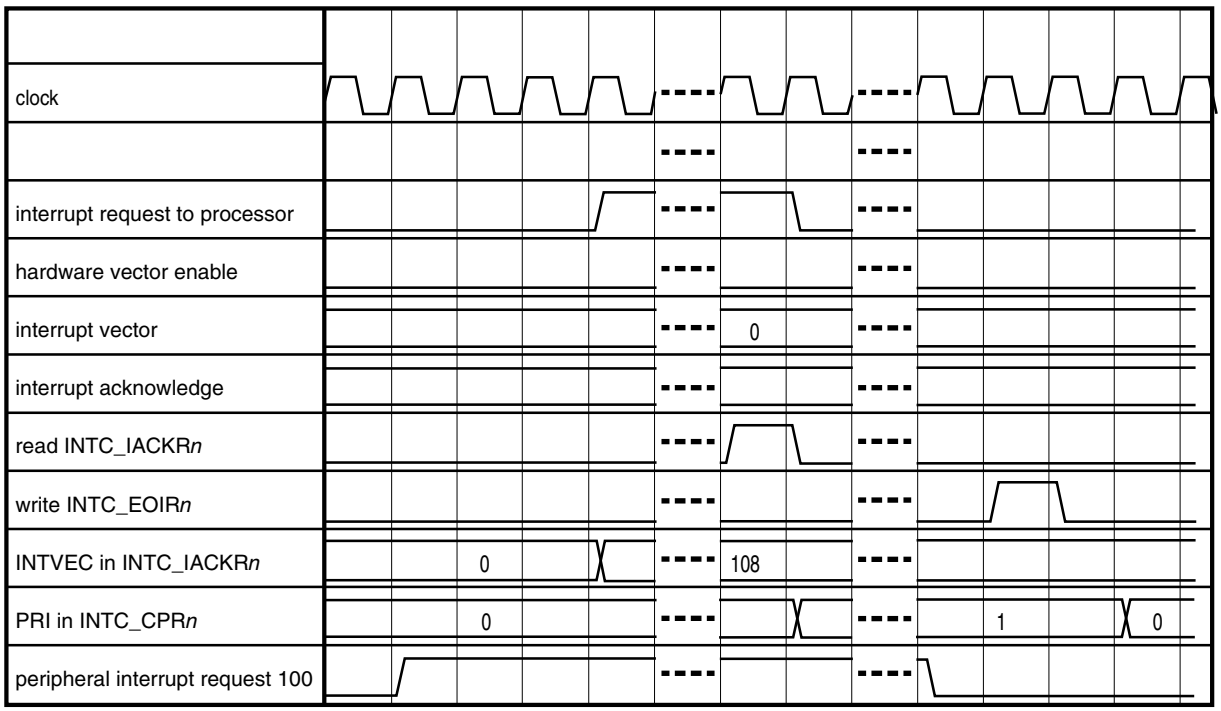


Figure 23-3. Timing diagram of software vector mode handshaking

23.7.3.2 Hardware vector mode handshaking

A timing diagram of the interrupt request and acknowledge handshaking in hardware vector mode, along with the handshaking near the end of the interrupt exception handler, is shown in Figure 23-4. As in software vector mode, the INTC examines the peripheral and software-settable interrupt requests, and when it finds an asserted interrupt request with a higher priority than PRI in the associated INTC_CPR*n*, it asserts the interrupt request to the associated processor. The INTVEC field in the associated INTC_IACKR*n* is updated with the preempting peripheral or software-settable interrupt request's vector when the interrupt request to the processor is asserted. The INTVEC field retains that value until the next time the interrupt request to the associated processor is asserted. In addition, the value of the interrupt vector to the associated processor matches the value of the INTVEC field in the associated INTC_IACKR*n*. The rest of the handshaking is described in Hardware vector mode.

The handshaking near the end of the interrupt exception handler, that is the writing to the associated INTC_EOIR*n*, is the same as in software vector mode. See End of interrupt exception handler.

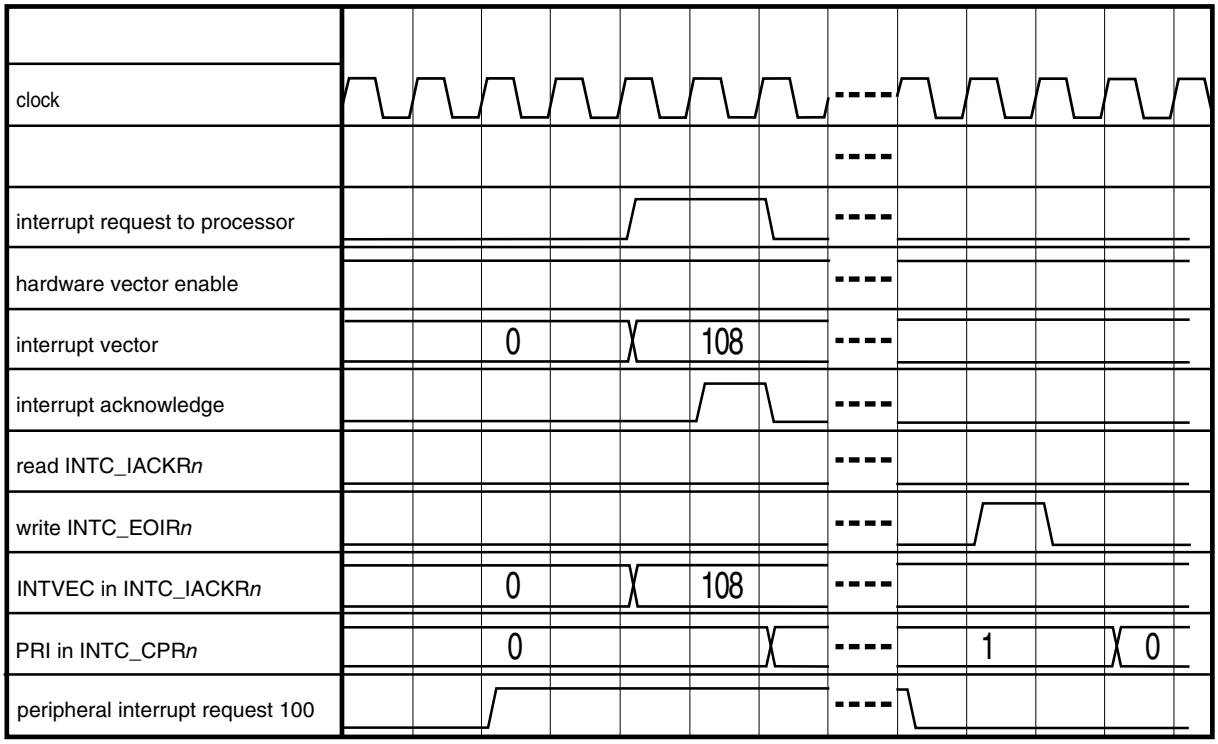


Figure 23-4. Timing diagram for hardware vector mode handshaking

23.8 Initialization/application information

This section contains initialization/application information.

23.8.1 Initialization flow

After exiting reset, all of the PRI*n* and PRC_SEL*n* fields in INTC_PSR*n* are zero, and PRI in the INTC_CPRn registers is 15. These reset values will prevent the INTC from asserting the interrupt request to the processors. The enable or mask bits in the peripherals are reset such that the peripheral interrupt requests are negated. An initialization sequence for allowing the peripheral and software-settable interrupt requests to cause an interrupt request to the processor is:

- interrupt_request_initialization:
  - Configure HVENn in INTC_BCR.
  - Configure VTBAn in INTC_IACKRn.
  - Raise the PRIn fields and set the PRC_SELn fields to the desired processor in INTC_PSRn.
  - Set the enable bits or clear the mask bits for the peripheral interrupt requests.

- Lower PRI in INTC_CPRn to zero.
- Enable processor(s) recognition of interrupts.

23.8.2 Interrupt exception handler

These example interrupt exception handlers excerpts use Power Architecture assembly code.

23.8.2.1 Software vector mode

In software vector mode for Power Architecture, there are 16 bytes of vector space available at the single ISR entry point.

```
interrupt_exception_handler:
b        interrupt_exception_handler_continued# 16 bytes available, branch to continue

interrupt_exception_handler_continued:
code to create stack frame, save working register, and save SRR0 and SRR1 (Power
Architecture Save/Restore registers) (not shown)

lis        r3,INTC_IACKRn@ha        # form adjusted upper half of INTC_IACKRn address
lwz        r3,INTC_IACKRn@l(r3)     # load INTC_IACKRn, which clears request to processor
lwz        r3,0x0(r3)                # load address of ISR from vector table
wrteei    1                        # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (Embedded Binary
Application Interface) (not shown)

mtlr    r3                         # move the INTC_IACKRn address into the link register
blrl                              # branch to ISR; link register updated with epilog
                                  # address
epilog:
code to restore most of context required by Power Architecture EABI (not shown)
# Popping the LIFO after the restoration of most of the context and the disabling of
processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                              # ensure store to clear flag bit has completed
lis        r3,INTC_EOIRn@ha        # form adjusted upper half of INTC_EOIRn address
li        r4,0x0                   # form 0 to write to INTC_EOIRn
wrteei    0                        # disable processor recognition of interrupts
stw        r4,INTC_EOIRn@l(r3)        # store to INTC_EOIRn, informing INTC to lower priority
code to restore SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

vector_table_base_address:
address of ISR for interrupt with vector 0
address of ISR for interrupt with vector 1
        .
        .
        .
address of ISR for interrupt with vector 1022
address of ISR for interrupt with vector 1023
ISRn:
code to service the interrupt event (not shown)
```

```
code to clear flag bit which drives interrupt request to INTC (not shown)
blr                                 # return to epilog
```

23.8.2.2 Hardware vector mode

This interrupt exception handler is useful with processor and system bus implementations that support a hardware vector.

```
interrupt_exception_handlern:
b        interrupt_exception_handler_continuedn# 16 bytes available, branch to continue

interrupt_exception_handler_continuedn:
code to create stack frame, save working register, and save SRR0 and SRR1 (not shown)

wrteei    1                    # enable processor recognition of interrupts

code to save rest of context required by Power Architecture EABI (not shown)

bl        ISRn                # branch to ISR for interrupt with vector n

epilog:
code to restore most of context required by Power Architecture EABI (not shown)

# Popping the LIFO after the restoration of most of the context and the disabling of
processor
# recognition of interrupts eases the calculation of the maximum stack depth at the cost of
# postponing the servicing of the next interrupt request.
mbar                             # ensure store to clear flag bit has completed
lis        r3,INTC_EOIRn@ha        # form adjusted upper half of INTC_EOIRn address
li        r4,0x0                # form 0 to write to INTC_EOIRn
wrteei    0                    # disable processor recognition of interrupts
stw        r4,INTC_EOIRn@l(r3)    # store to INTC_EOIRn, informing INTC to lower priority

SRR0 and SRR1, restore working registers, and delete stack frame (not shown)

rfi

ISRn:
code to service the interrupt event
code to clear flag bit which drives interrupt request to INTC

blr        # branch to epilog
```

23.8.3 ISR, RTOS, and task hierarchy

The RTOS and all of the tasks under its control typically execute with PRI in INTC_CPR*n* having a value of 0. The RTOS will execute the tasks according to whatever priority scheme it may have, but that priority scheme is independent and has a lower priority of execution than the priority scheme of the INTC. In other words, the ISRs execute above INTC_CPR*n* priority 0 and outside the control of the RTOS, the RTOS executes at INTC_CPR*n* priority 0, and while the tasks execute at different priorities under the control of the RTOS, they also execute at INTC_CPRn priority 0.

If a task shares a resource with an ISR and the Priority Ceiling Protocol (PCP) is being used to manage that shared resource, then the task's priority can be elevated in the INTC_CPR*n* while the shared resource is being accessed.

An ISR whose PRI*n* in INTC_PSR*n* has a value of 0 will not cause an interrupt request to the selected processor, even if its peripheral or software-settable interrupt request is asserted. For a peripheral interrupt request, not setting its enable bit or disabling the mask bit will cause it to remain negated, which consequently also will not cause an interrupt request to the processor. Since the ISRs are outside the control of the RTOS, this ISR will not run unless called by another ISR or the interrupt exception handler, perhaps after executing another ISR.

23.8.4 Order of execution

An ISR with a higher priority can preempt an ISR with a lower priority, regardless of the unique vectors associated with each of their peripheral or software-settable interrupt requests. However, if multiple peripheral or software-settable interrupt requests are asserted, more than one of these interrupt requests has the highest priority and that priority is high enough to cause preemption, the INTC selects the one with the lowest unique vector regardless of the order in time that they asserted. However, the ability to meet deadlines with this scheduling scheme is no less than if the ISRs execute in the time order that their peripheral or software-settable interrupt requests asserted.

The example in the following table shows the order of execution of both cases, ISRs with different priorities and ISRs with the same priority.

Table 23-3. Order of ISR execution example

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108 [1]	ISR208	ISR308	ISR408	interrupt exception handler	
1	RTOS at priority 0 is executing.	X						0
2	Peripheral interrupt request 100 at priority 1 asserts. Interrupt taken.		X					1
3	Peripheral interrupt request 400 at priority 4 asserts. Interrupt taken.					X		4
4	Peripheral interrupt request 300 at priority 3 asserts.					X		4
5	Peripheral interrupt request 200 at priority 3 asserts.					X		4
6	ISR408 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1

Table continues on the next page...

Table 23-3. Order of ISR execution example (continued)

Step #	Step Description	Code executing at end of step						PRI in INTC_CPR at end of step
		RTOS	ISR108 [1]	ISR208	ISR308	ISR408	interrupt exception handler	
7	Interrupt taken. ISR208 starts to execute, even though peripheral interrupt request 300 asserted first.			X				3
8	ISR208 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
9	Interrupt taken. ISR308 starts to execute.				X			3
10	ISR308 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	1
11	ISR108 completes. Interrupt exception handler writes to INTC_EOIR*n*.						X	0
12	RTOS continues execution.	X						0

1. ISR108 executes for peripheral interrupt request 100 because the first eight ISRs are for software-settable interrupt requests.

23.8.5 Priority ceiling protocol

This section describes the priority ceiling protocol.

23.8.5.1 Elevating priority

The PRI field in INTC current priority register (INTC_CPR*n*) is elevated in the OSEK PCP to the ceiling of all of the priorities of the ISRs that share a resource. This protocol therefore allows coherent accesses of the ISRs to that shared resource.

For example, ISR1 has a priority of 1, ISR2 has a priority of 2, and ISR3 has a priority of 3. They all share the same resource. Before ISR1 or ISR2 can access that resource, they must raise the PRI value in INTC_CPR*n* to 3, the ceiling of all of the ISR priorities. After they release the resource, they must lower the PRI value in INTC_CPR*n* to prevent further priority inversion. If they do not raise their priority, then ISR2 can preempt ISR1, and ISR3 can preempt ISR1 or ISR2, possibly corrupting the shared resource. Another possible failure mechanism is deadlock if the higher priority ISR needs the lower priority ISR to release the resource before it can continue, but the lower priority ISR cannot release the resource until the higher priority ISR completes and execution returns to the lower priority ISR.

Using the PCP instead of disabling processor recognition of all interrupts eliminates the time when accessing a shared resource that all higher priority interrupts are blocked. For example, while ISR3 cannot preempt ISR1 while it is accessing the shared resource, all of the ISRs with a priority higher than 3 can preempt ISR1.

23.8.5.2 Ensuring coherency

This section discusses how to ensure coherency.

23.8.5.2.1 Interrupt with blocked priority

A scenario can exist that can cause non-coherent accesses to the shared resource. As an example, ISR1 and ISR2 both share a resource. ISR1 has a lower priority than ISR2. ISR1 is executing, and it writes to the INTC_CPR*n*. The instruction following this store is a store to a value in a shared coherent data block. Either just before or at the same time as the first store, the INTC asserts the interrupt request to the processor because the peripheral interrupt request for ISR2 has asserted. As the processor is responding to the interrupt request from the INTC, and as it is aborting transactions and flushing its pipeline, it is possible (in some implementations) that both of these stores are executed. ISR2 thereby assumes that it can access the data block coherently, but the data block has been corrupted.

OSEK uses the GetResource and ReleaseResource system services to manage access to a shared resource. To prevent this corruption of a coherent data block, modifications to PRI in INTC_CPR*n* can be made by those system services with the code sequence:

```
GetResource:
        wrteei 0                # disable external interrupts to the Processor
        raise PRI                # Write to CPR, cache inhibit, guarded
        mbar                    # flush out writes from store buffer
        wrteei 1                # enable external interrupts to the Processor
        isync                    # re-fetch Processor pipeline

ReleaseResource:
        mbar                    # flush out writes from store buffer
        wrteei 0                # disable external interrupts to the Processor
        lower PRI                # Write to CPR, cache inhibit, guarded
        wrteei 1                # enable external interrupts to the Processor
```

23.8.5.2.1.1 Interrupt request to processor

Referencing Figure 23-1, the interrupt request logic to the processor is shown in the following figure.

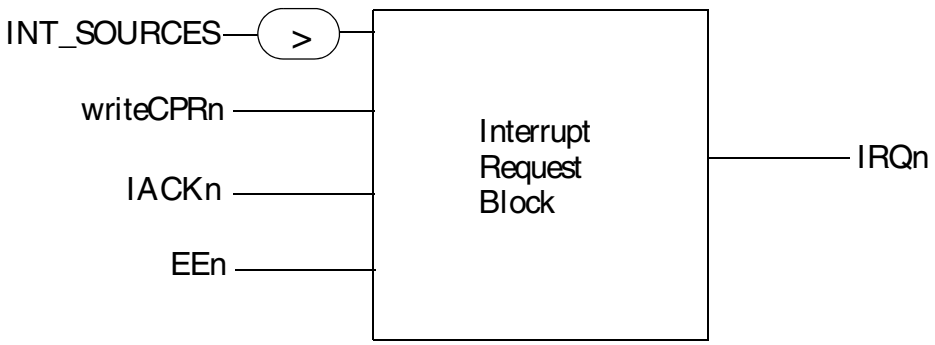


Figure 23-5. Interrupt Request Block Diagram

Assuming that there are no IRQn deasserted (no pending interrupt requests to processor n), if one of the external interrupt sources (INT_SOURCES) has a priority higher than the INTC_CPRn, then interrupt request to the processor (IRQn) is asserted. It will stay asserted until one of the following conditions is true:

- Interrupt acknowledge (IACK) is asserted
- If the EEn bit has been cleared by the wrteei instruction (see the code sequence in Interrupt with blocked priority), IRQn will be re-evaluated when processor n writes to the INTC_CPRn while processor n ignores IRQn

This provides a safe way to guarantee that no interrupts will be recognized by processor n while updating the INTC_CPRn.

23.8.5.2.2 Raised priority preserved

Before the instruction after the GetResource system service executes, all pending transactions have completed. These pending transactions can include an ISR for a peripheral or software-settable interrupt request whose priority was equal to or lower than the raised priority. The shared coherent data block now can be accessed coherently. The following figure shows the timing diagram for this scenario, and the following table explains the events. The example is for software vector mode. Except for the method of retrieving the vector and acknowledging the interrupt request to the processor, hardware vector mode is identical.

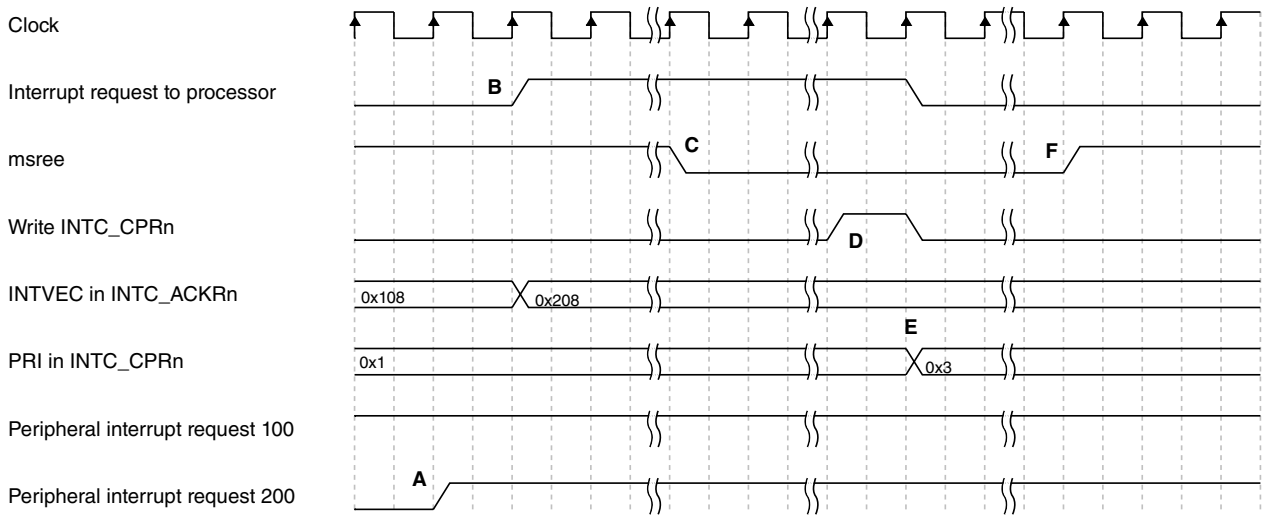


Figure 23-6. Timing diagram of raised priority preserved

Table 23-4. Raised priority preserved events

Event	Description
A	Peripheral interrupt request 200 of priority 2 asserts during execution of ISR108 running at priority 1.
B	Interrupt request to processor asserts. INTVEC in INTC_IACKRn updates with vector for that peripheral interrupt request.
C	Write to msree bit to disable external interrupts
D	ISR108 writes to INTC_CPRn to raise priority to 3 before accessing shared coherent data block.
E	PRI in INTC_CPRn now at 3, reflecting the write. This write, just before accessing data block, is the last instruction the processor executes before being interrupted.
F	Write to msree bit to enable external interrupts

23.8.6 Selecting priorities according to request rates and deadlines

The selection of the priorities for the ISRs can be made using Rate Monotonic Scheduling (RMS) or a superset of it, Deadline Monotonic Scheduling (DMS). In RMS, the ISRs which have higher request rates have higher priorities. In DMS, if the deadline is before the next time the ISR is requested, then the ISR is assigned a priority according to the time from the request for the ISR to the deadline, not from the time of the request for the ISR to the next request for it.

For example, ISR1 executes every 100 µs, ISR2 executes every 200 µs, and ISR3 executes every 300 µs. ISR1 has a higher priority than ISR2 which has a higher priority than ISR3. However, if ISR3 has a deadline of 150 µs, then it has a higher priority than ISR2.

The INTC supports 16 priorities, which may be many fewer than the number of ISRs. In this case, the ISRs should be grouped with other ISRs that have similar deadlines. For example, a priority could be allocated for every time the request rate doubles. ISRs with request rates around 1 ms would share a priority, ISRs with request rates around 500 µs would share a priority, ISRs with request rates around 250 µs would share a priority, and so on. With this approach, a range of ISR request rates of 2^{16} could be covered, regardless of the number of ISRs.

Reducing the number of priorities does cause some priority inversion, which reduces the processor's ability to meet its deadlines. However, reducing the number of priorities can reduce the size and latency through the interrupt controller. It also allows easier management of ISRs with similar deadlines that share a resource. They can be placed at the same priority without any further priority inversion, and they do not need to use the PCP to access the shared resource.

23.8.7 Software-settable interrupt requests

The software-settable interrupt requests can be used in two ways. They can be used to schedule a lower priority portion of an ISR and for processors to interrupt other processors in a multiple processor system.

23.8.7.1 Scheduling a lower priority portion of an ISR

A portion of an ISR needs to be executed at the PRIn value in INTC_PSR*n*, which becomes the PRI value in INTC_CPR*n* with the interrupt acknowledge. The ISR, however, can have a portion of it which does not need to be executed at this higher priority. Therefore, executing this later portion which does not need to be executed at this higher priority can prevent the execution of ISRs which do not have a higher priority than the earlier portion of the ISR but do have a higher priority than the later portion of the ISR needs. This preemptive scheduling inefficiency reduces the processor's ability to meet its deadlines.

One option is for the ISR to complete the earlier higher priority portion, but then schedule through the RTOS a task to execute the later lower priority portion. However, some RTOSs can require a large amount of time for an ISR to schedule a task. Therefore, a second option is for the ISR, after completing the higher priority portion, to set a SET bit

in INTC_SSCIR*n*. Writing a '1' to SET causes a software-settable interrupt request. This software-settable interrupt request, which usually will have a lower PRI*n* value in the INTC_PSR*n*, therefore will not cause preemptive scheduling inefficiencies.

23.8.7.2 Scheduling an ISR on another processor

Since the SET bits in the INTC_SSCIR*n* are memory mapped, processors in multiple processor systems can schedule ISRs on the other processors. One possible application is if one processor simply wants to command another processor to perform a piece of work, and the initiating processor does not need to use the results of that work. If the initiating processor needs to know if the processor executing the software-settable ISR has not completed the work before asking it to again execute that ISR, it can check if the corresponding CLR bit in INTC_SSCIR*n* is asserted before again writing a 1 to the SET bit.

Another application is the sharing of a block of data. For example, a first processor has completed accessing a block of data and wants a second processor to then access it. Furthermore, after the second processor has completed accessing the block of data, the first processor again wants to access it. The accesses to the block of data must be done coherently. The procedure is that the first processor writes a 1 to a SET bit on the second processor. The second processor, after accessing the block of data, clears the corresponding CLR bit and then writes 1 to a SET bit on the first processor, informing it that it now can access the block of data.

23.8.8 Lowering priority within an ISR

In implementations without the software-settable interrupt requests in INTC_SSCIR*n*, one way (besides scheduling a task through an RTOS) to prevent preemptive scheduling inefficiencies with an ISR whose work spans multiple priorities (as described in Scheduling a lower priority portion of an ISR) is to lower the current priority. However, the INTC has a LIFO whose depth is determined by the number of priorities.

Note

Lowering the PRI value in INTC_CPR*n* within an ISR to below the ISR's corresponding PRI value in INTC_PSR*n* allows more preemptions than the depth of the LIFO can support.

Therefore, through its use of the LIFO the INTC does not support lowering the current priority within an ISR as a way to avoid preemptive scheduling inefficiencies.

23.8.9 Negating an interrupt request outside of its ISR

This section discusses the negation of an interrupt service request outside of its ISR.

23.8.9.1 Negating an interrupt request as a side effect of an ISR

Some peripherals have flag bits which can be cleared as a side effect of servicing a peripheral interrupt request. For example, reading a specific register can clear the flag bits, and consequently their corresponding interrupt requests too. This clearing as a side effect of servicing a peripheral interrupt request can cause the negation of other peripheral interrupt requests besides the peripheral interrupt request whose ISR presently is executing. This negating of a peripheral interrupt request outside of its ISR can be a desired effect.

23.8.9.2 Negating multiple interrupt requests in one ISR

An ISR can clear other flag bits besides its own flag bit. One reason that an ISR clears multiple flag bits is because it serviced those other flag bits, and therefore the ISRs for these other flag bits do not need to be executed.

23.8.9.3 Proper setting of interrupt request priority

Whether an interrupt request negates outside of its own ISR due to the side effect of an ISR execution or the intentional clearing of a flag bit, the priorities of the peripheral or software-settable interrupt requests for these other flag bits must be selected properly. Their PRI*n* values in INTC_PSR*n* must be selected to be at or lower than the priority of the ISR that cleared their flag bits. Otherwise, those flag bits still can cause the interrupt request to the processor to assert. Furthermore, the clearing of these other flag bits also has the same timing relationship to the writing to INTC_SSCIR*n* as the clearing of the flag bit that caused the present ISR to be executed. Refer to End of interrupt exception handler for more information.

A flag bit whose enable bit or mask bit is negating its peripheral interrupt request can be cleared at any time, regardless of the peripheral interrupt request's PRI*n* value in INTC_PSR*n*.

23.8.10 Examining LIFO contents

There are multiple methods for tracking interrupts in a system, one of which is described here using existing hardware (LIFO) within the INTC. Although the LIFO contents are not memory-mapped, the user can read the contents by popping the LIFO and reading the PRI field in the INTC current priority register (INTC_CPR*n*). To avoid a lower-level interrupt being serviced because the popping of the LIFO updates the INTC_CPR*n*, the processor recognition of interrupts should be disabled when examining LIFO contents. The pseudo-code is as follows:

```
      wrteei 0                # disable processor recognition of external interrupts
      oldCPRn = INTC_CPRn;    # save INTC_CPRn
      (branch to the pop_lifo)

pop_lifo:
      store INTC_EIORn        # pop INTC_CPR from LIFO, examine PRI, etc...
      examine INTC_CPRn[PRI], and store onto stack
      if PRI is not zero or value when interrupts were enabled, branch to pop_lifo
      branch to push_lifo
```

When the examination is complete, the LIFO can be restored using this code sequence:

```
push_lifo:
      load stacked PRI value and store to INTC_CPRn
      load INTC_IACKn         # IACK; push INTC_CPRn into LIFO
      if stacked PRI values are not depleted, branch to push_lifo

      INTC_CPRn    = oldCPRn; # restore original INTC_CPRn and re-evaluate pending external
interrupts
      wrteei 1                # enable processor recognition of interrupts
```

NOTE

Disabling the processor recognition of interrupts during LIFO examination will introduce latency, but none of the interrupt requests will be missed.

23.9 Interrupt sources

The list of interrupt sources is chip-specific. For this list, see the chip-specific INTC information.

Chapter 24
Wakeup Unit (WKPU)

24.1 Chip-specific WKPU information

24.1.1 Wakeup configuration

The Wakeup Unit supports 2 internal sources and 30 external sources that can generate interrupts or wakeup events. it also supports a non-maskable interrupt input. The following table shows the WKPU configuration in the device.

Feature	Value
Number of NMI sources	1
Number of interrupt sources	2 - internal, 30 - external
Number of glitch filters for external interrupts	30
Number of external interrupt vectors	4

24.1.2 NMI configuration

The following figure shows the WKUP NMI configuration implemented on the device. The Wakeup Unit (WKPU) supports one external source that can cause non-maskable interrupts to on-chip cores and wakeup events to the system. This NMI external source can also generate a functional reset event to the MC_RGM.

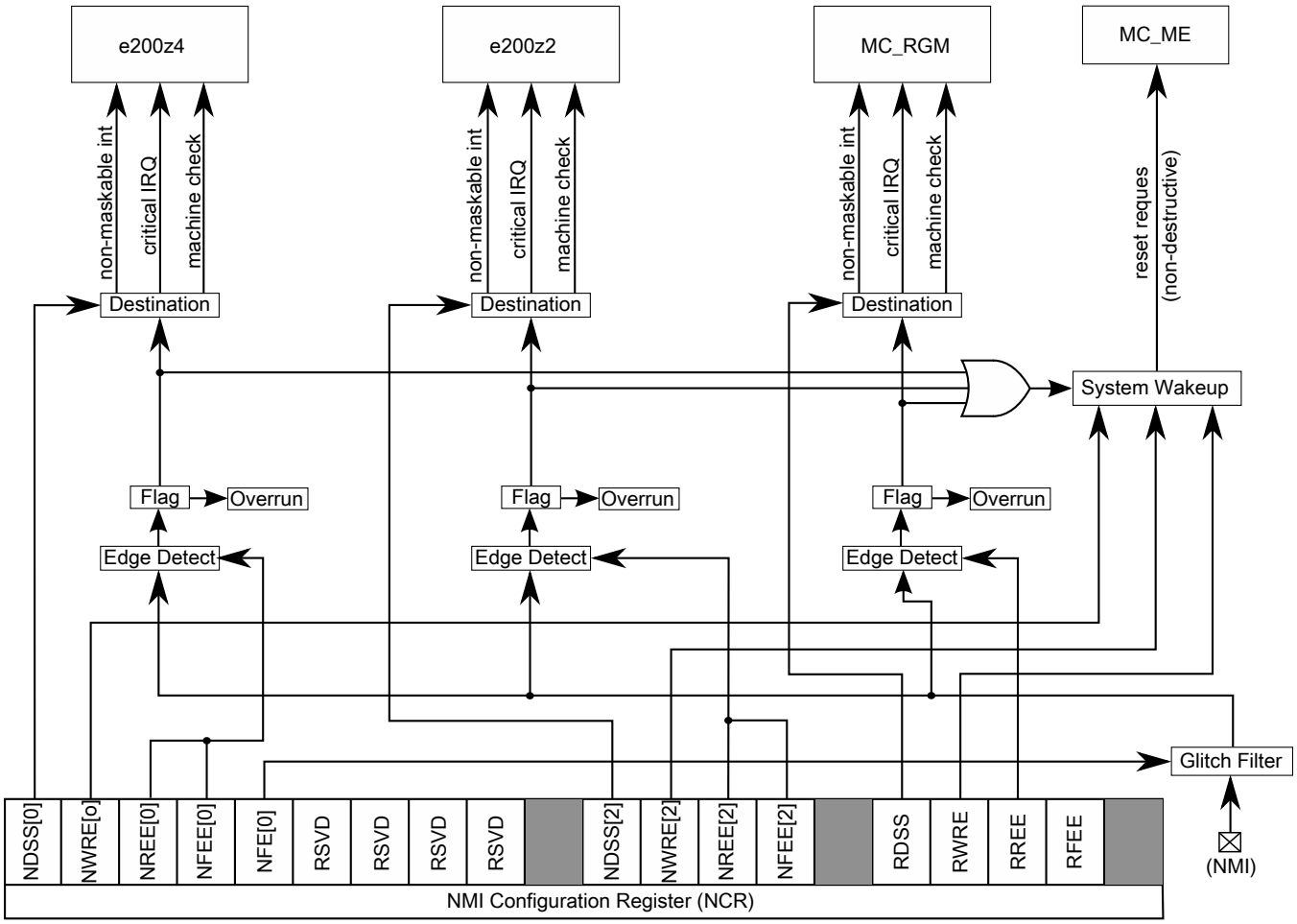


Figure 24-1. Wakeup NMI configuration

24.1.3 Interrupts

The WKPU allows the external NMI pin to assert the non-maskable interrupts of the cores on the device. These interrupts include non-maskable interrupts, critical interrupts, or machine check requests to any of the cores (excluding HSM).

- The NMI destination interrupt is controlled by the user through the configuration of the NDSS bits.
- The enabling of rising edge, falling edge, or either edge reaction to the NMI pin is independent for each core, and is controlled by NFEE[2:0] and NREE[2:0] bits in the NCR register.
- The WKPU supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event.
- The NMI supports a status flag and an overrun flag which are located in the NSR register.

The routing of External wakeup/interrupts with the WKPU and system interrupt controller is shown below.

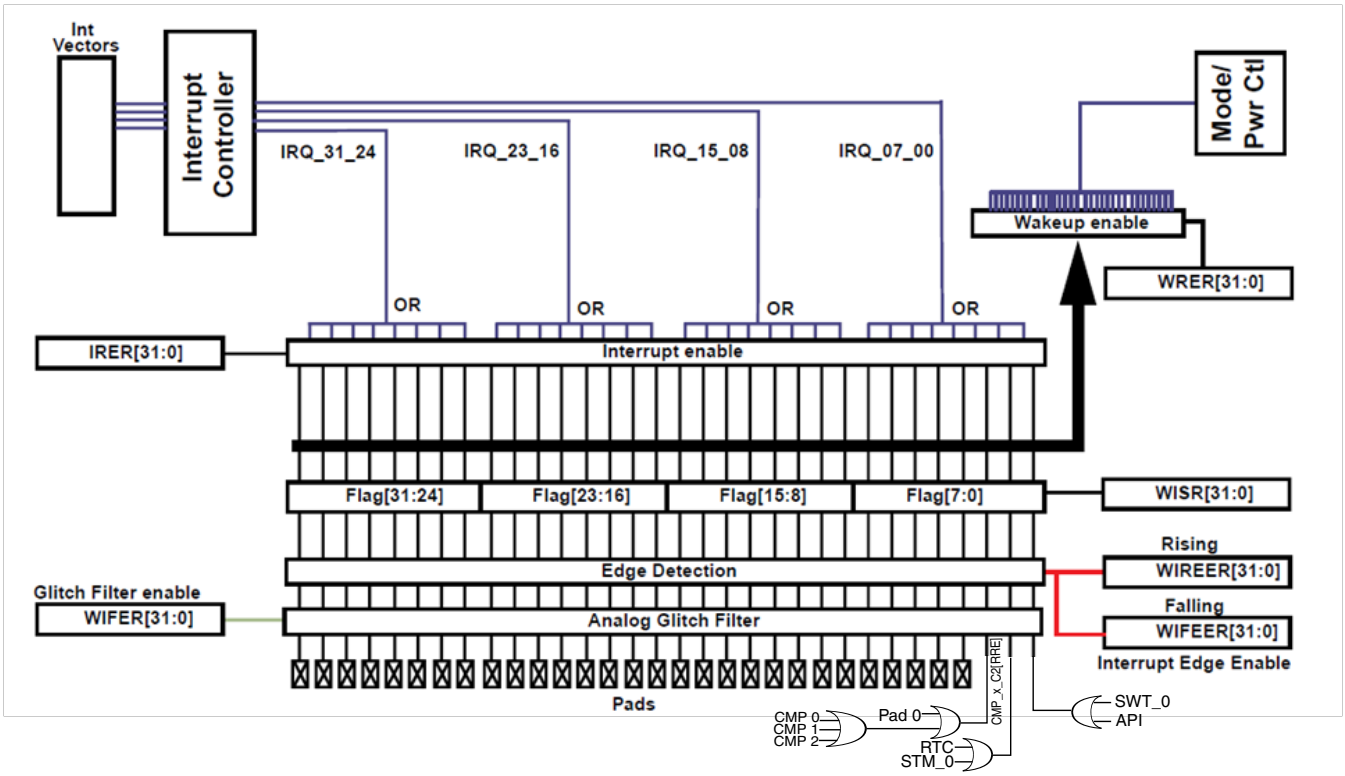


Figure 24-2. WKPU External interrupts connectivity

- The SWT0 and API inputs provides a wakeup trigger to WKPU interrupt source 0. API wakeup is supported only when all the CMP_x_C2 [RRE] fields are disabled.
- The RTC and STM_0 provides a wakeup trigger to WKPU interrupt source 1.

If the user configures any or all of the CMP_x_C2 [RRE] fields to be active then WKPU interrupt source 3 should be dedicated to interrupt (generated in STANDBY/STOP or LPU_STANDBY/LPU_STOP) from any of the CMP_x_C2 [RRE] fields. If the user is not using any of the CMP then it is desirable that the pad may still operate as a digital WKPU input signal. Thus, users not taking advantage of the CMP do not lose a digital wakeup source.

NOTE

Software must enable the WKPU (by using the appropriate MC_ME_PCTLx register for the WKPU) before entering any low-power mode of the device.

24.1.4 Wakeup Sources for various modes

All peripheral interrupts can cause wakeup from STOP mode if they support STOP mode and are enabled via ME_PCTL and system clock is available. LPU Peripheral interrupts can cause wakeup from LPU SLEEP mode if they are enabled via LPU MDIS. The peripherals which can cause wakeup from LPU mode are covered in the table below. The asynchronous interrupts can cause wake up from low power modes when system clock is gated.

The below table lists the asynchronous wakeup sources in low power modes.

Table 24-1. Asynchronous wakeup sources in low power modes

Sources	STOP	LPU_STOP	STANDBY	LPU_STANDBY
SWT_0	√	√	√[1]	√[1]
SWT_1	√	.	.	.
SWT_HSM	√	.	.	.
STM_0	√	√	√[1]	√[1]
STM_1	√	.	.	.
CMP 0, CMP 1 and CMP 2 (Continuous mode comparisons)	√	√	.	.
CMP 0, CMP 1 and CMP 2 (round robin mode comparisons)	√	√	√[1]	√[1]
FlexCAN_0 (Pretended Networking)	√	√	.	.
ENET 0 (magic packet detection)	√	.	.	.
RTI (timeout)	√	.	.	.
NMI	√[1]	√[1]	√[1]	√[1]
WKPU	√[1]	√[1]	√[1]	√[1]
RTC	√[1]	√[1]	√[1]	√[1]

1. This requires prior configuration in WKPU to generate asynchronous wakeup when SYSCLK is disabled.

24.1.5 NMI on standby exit

The NMI configuration register (NCR) would be:

1. Always reset in STANDBY mode
2. Reset along with Z2 core in LPU modes

Hence NMI from WKPU would never be generated on STANDBY exit until the SW is ready and has initialized the NCR register.

24.1.6 Register reset values

Some of the registers in this module have chip-specific reset values. They are listed below.

Register	Reset value
WKPU_NCR	0x6060_6060

24.1.7 Pull control of wakeup pads

The following table mentions how the PULL value of the WKPU pads are controlled by SIUL and WKPU configuration.

Table 24-2. Pull control of wakeup pads

Device Mode	SIUL2_MSCRn[PUE]	SIUL2_MSCRn[PUS]	WKPU_WIPUER[IPUE]	Affect on WKPU PAD
All except STANDBY modes	1	0	x	Pull down
	1	1	x	Pull up
	0	x	0	Highz
	0	0	1	Pull down
	0	1	1	Pull up

Table 24-3. Pull control of wakeup pads

Device Mode	State of SIUL2_MSCRn[PUS] before entering Standby	WKPU_WIPUER[IPUE]	Affect on WKPU PAD
STANDBY/LPU_STANDBY mode	x	0	Highz
	0	1	Pull down
	1	1	Pull up

The pull up/down voltage level for unconnected wakeup/interrupt input is VDD_HV_A supply level.

Recommended WKPU Configuration before entering into STANDBY/LPU_STANDBY mode

- Disable IBE of all EIRQ pads in SIUL2 Multiplexed Signal Configuration Register (SIUL2_MSCR*n*).
- CLEAR the Wakeup/Interrupt Status Flag Register (WKPU_WISR) by writing W1C to all bits.

- ENABLE wakeup for desired sources in Wakeup Request Enable Register (WKPU_WRER).
- ENABLE edge selection in Wakeup/Interrupt Rising-Edge Event Enable Register (WKPU_WIREER) or Wakeup/Interrupt Falling-Edge Event Enable Register (WKPU_WIFEER) as per the wakeup sources enabled in Wakeup Request Enable Register (WKPU_WRER) configuation.

24.2 Introduction

The WKPU supports external sources that can generate interrupts or wakeup events and external source(s) that can cause non-maskable interrupt request(s) or wakeup event(s). In addition, it combines its wakeup events with those generated from other wakeup sources to supply a single wakeup to the system. The block diagram of the WKPU and its interfaces to other system components is shown below.

NOTE

The signal widths in the following diagram might not depict this device's particular configuration. See the Chip-specific WKPU information for more information.

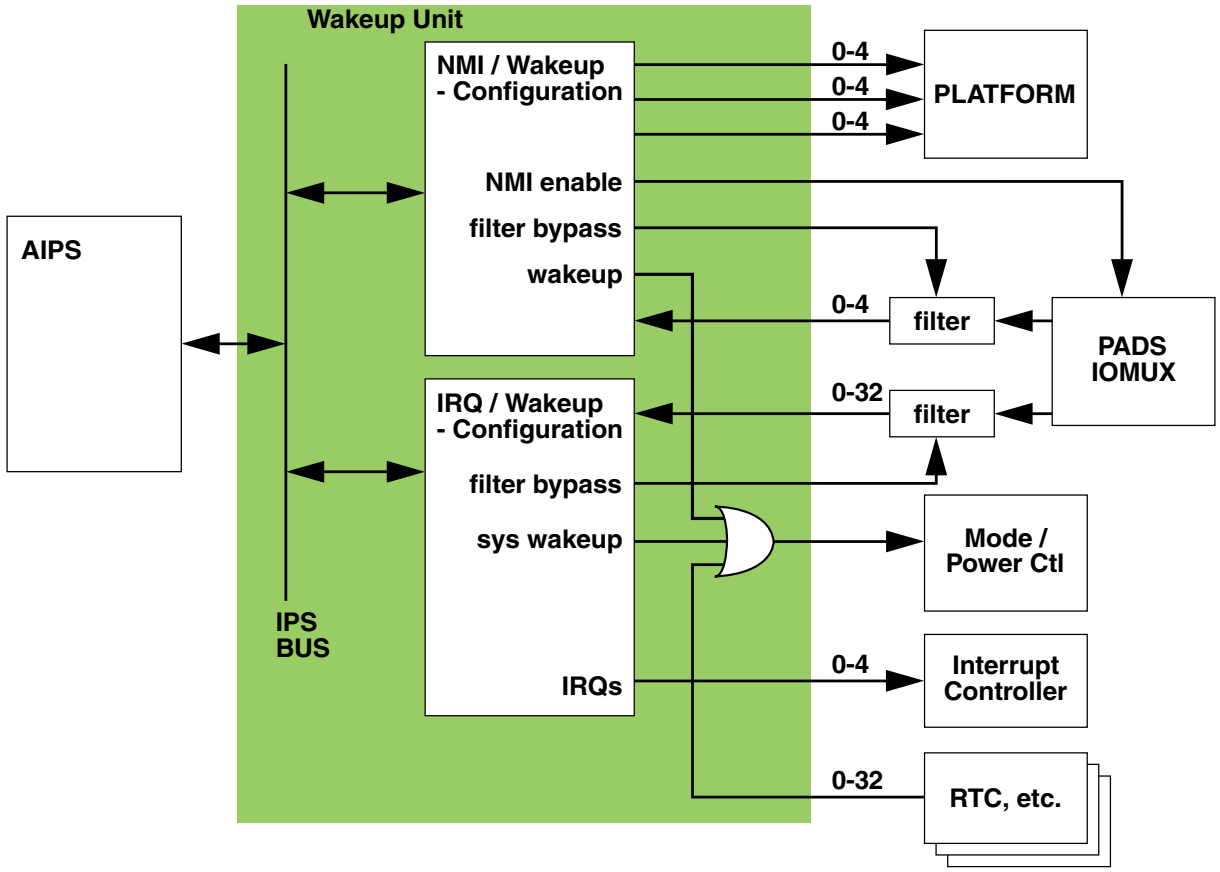


Figure 24-3. WKPU and connected system components

24.3 Features

The WKPU supports these features:

- Non-maskable Interrupt support with:
  - External NMI sources
  - Analog glitch filters
- Independent interrupt destination for each core:
  - Non-maskable interrupt
  - Critical interrupt
  - Machine check request
- Active edge selection control for events to each core
- Configurable system wakeup triggering from NMI source(s)

24.4 External signal description

The WKPU has signal inputs that can be used as external interrupt sources in normal run mode or as system wakeup sources in certain power down modes.

24.5 WKPU memory map and register definition

This section provides a detailed description of all registers accessible in the WKPU module.

NOTE

Reserved registers will read as 0, writes will have no effect. Transfer error will be generated when trying to access completely reserved register space. The field length in external pad control registers depends on the number of WKPU channels implemented in a chip.

WKPU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	NMI Status Flag Register (WKPU_NSR)	32	w1c	0000_0000h	24.5.1/570
8	NMI Configuration Register (WKPU_NCR)	32	R/W	See section	24.5.2/572
14	Wakeup/Interrupt Status Flag Register (WKPU_WISR)	32	R/W	0000_0000h	24.5.3/575
18	Interrupt Request Enable Register (WKPU_IRER)	32	R/W	0000_0000h	24.5.4/576
1C	Wakeup Request Enable Register (WKPU_WRER)	32	R/W	0000_0000h	24.5.5/577
28	Wakeup/Interrupt Rising-Edge Event Enable Register (WKPU_WIREER)	32	R/W	0000_0000h	24.5.6/577
2C	Wakeup/Interrupt Falling-Edge Event Enable Register (WKPU_WIFEER)	32	R/W	0000_0000h	24.5.7/578
30	Wakeup/Interrupt Filter Enable Register (WKPU_WIFER)	32	R/W	0000_0000h	24.5.8/578
34	Wakeup/Interrupt Pull Enable Register (WKPU_WIPER)	32	R/W	0000_0000h	24.5.9/579

24.5.1 NMI Status Flag Register (WKPU_NSR)

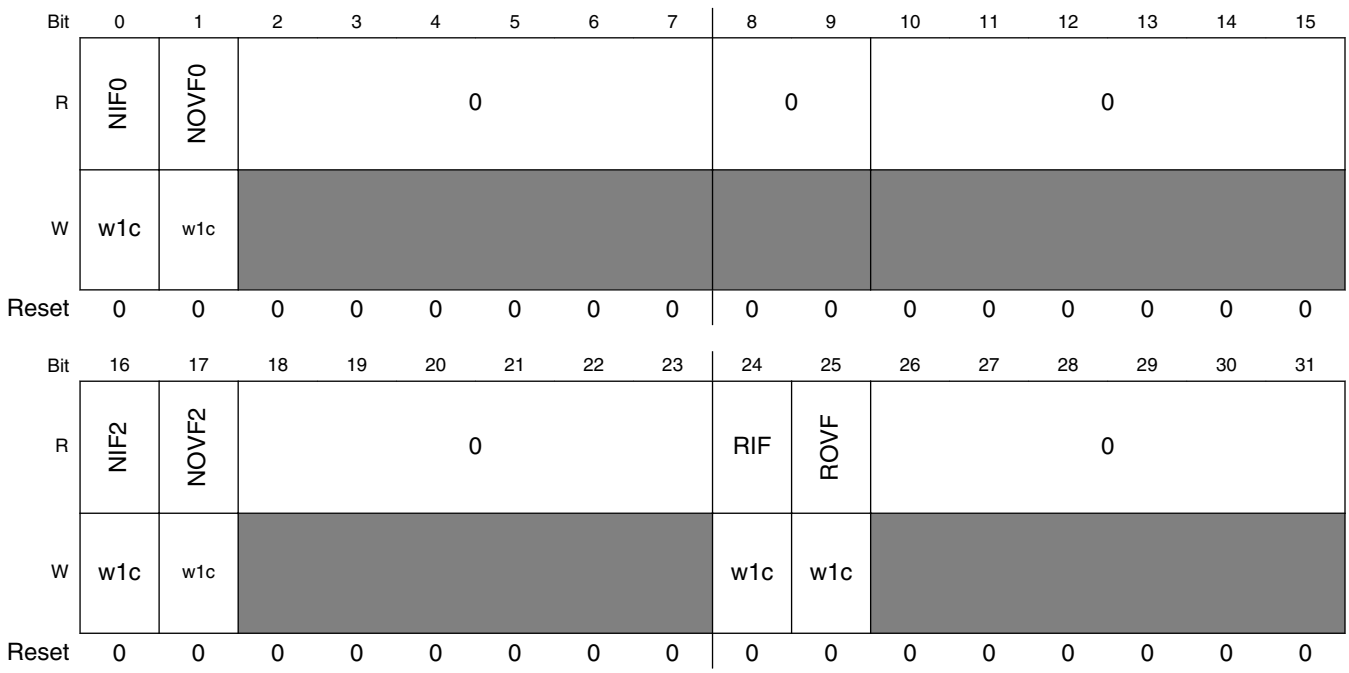
This register holds the non-maskable interrupt status flags.

NOTE

This register is accessible by 8-, 16-, and 32-bit read/write operations.

Access: User read/write

Address: 0h base + 0h offset = 0h



WKPU_NSR field descriptions

Field	Description
0 NIF0	NMI Status Flag 0. This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE0 or NFEE0 set), NIF0 causes an interrupt request. 0 No event has occurred on the pad 1 An event as defined by NREE0 and NFEE0 has occurred
1 NOVF0	NMI Overrun Status Flag 0. This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF0 value whenever a NMI event occurs, thereby indicating to the software that a NMI occurred while the last one was not yet serviced. If enabled (NREE0 or NFEE0 set), NOVF0 causes an interrupt request. 0 No overrun has occurred on NMI input 0 1 An overrun has occurred on NMI input 0
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 NIF2	NMI Status Flag 2. This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (NREE2 or NFEE2 set), NIF2 causes an interrupt request.

Table continues on the next page...

WKPU_NSR field descriptions (continued)

Field	Description
	0 No event has occurred on the pad 1 An event as defined by NREE2 and NFEE2 has occurred
17 NOVF2	NMI Overrun Status Flag 2. This flag can be cleared only by writing a 1. Writing a 0 has no effect. It will be a copy of the current NIF2 value whenever a NMI event occurs, thereby indicating to the software that a NMI occurred while the last one was not yet serviced. If enabled (NREE2 or NFEE2 set), NOVF2 causes an interrupt request. 0 No overrun has occurred on NMI input 2 1 An overrun has occurred on NMI input 2
18–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RIF	Reset Status Flag If enabled (i.e., if NCR[RREE] or NCR[RFEE] is set), this status bit is set and will cause a functional reset request to the RGM if NCR[RDSS] 0b00. This bit is automatically cleared if the ESR1 reset request is configured in the MC_RGM to generate a long or short “functional” reset. 0 No event has occurred on the pad 1 An event as defined by NCR[RREE] and NCR[RFEE] has occurred
25 ROVF	Reset Overrun Status Flag This field is a copy of the current RIF value whenever a NMI event occurs, thereby indicating to the software that an NMI occurred while the last one was not yet serviced. If enabled (i.e., NCR[RREE] or NCR[RFEE] is set), this field can cause a reset request to MC_RGM if NCR[RDSS] 0b00. This bit is automatically cleared if the ESR1 reset request is configured in the MC_RGM to generate a long or short “functional” reset. 0 No overrun has occurred on the NMI reset request input 1 An overrun has occurred on the NMI reset request input
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

24.5.2 NMI Configuration Register (WKPU_NCR)

This register holds the configuration bits for the non-maskable interrupt settings.

NOTE

- This register is accessible by 8-, 16-, and 32-bit read/write operations.
- Writing a 0 to both NREE[*n*] and NFEE[*n*] disables the NMI functionality completely (i.e., no system wakeup or interrupt will be generated on any pad activity).

Access: User read/write

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	NLOCK0	NDSS0		NWRE0	0	NREE0	NFEE0	NFE0	0	0		0	0	0	0	Reserved
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NLOCK2	NDSS2		NWRE2	0	NREE2	NFEE2	Reserved	RLOCK	RDSS		RWRE	0	RREE	RFEE	Reserved
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- See the chip-specific WKPU information for the reset value of this register.

WKPU_NCR field descriptions

Field	Description
0 NLOCK0	NMI Configuration Lock Register 0. Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset . Writing a 0 has no effect.
1–2 NDSS0	NMI Destination Source Select 0. **NOTE:** If this field is configured for machine check exception, the following steps must be performed in order to receive the exception on the NMI Pad: 1. Set MSR[ME], Machine Check Enable field of the core's Machine State Register, to 1. 2. Set HID0[EMCP] to 1. 00 Non-maskable interrupt 01 Critical interrupt 10 Machine check request 11 No NMI, critical Interrupt, or machine check request generated.
3 NWRE0	NMI Wakeup Request Enable 0 **NOTE:** If wakeup requests are disabled, the corresponding NDSS field must be set to 11 to disable wakeups from an NMI, critical interrupt, or machine check. 0 System wakeup requests from the corresponding NIF0 bit are disabled. 1 A set NIF0 bit or set NOVF0 bit causes a system wakeup request.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 NREE0	NMI Rising-edge Events Enable 0. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
6 NFEE0	NMI Falling-edge Events Enable 0. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

Table continues on the next page...

WKPU_NCR field descriptions (continued)

Field	Description
7 NFE0	NMI Filter Enable 0. Enable analog glitch filter on the NMI pad input. 0 Filter is disabled 1 Filter is enabled
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 Reserved	This field is reserved. Always write 0 to this field.
16 NLOCK2	NMI Configuration Lock Register 2. Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset . Writing a 0 has no effect.
17–18 NDSS2	NMI Destination Source Select 2. **NOTE:** If this field is configured for machine check exception, the following steps must be performed in order to receive the exception on the NMI Pad: 1. Set MSR[ME], Machine Check Enable field of the core's Machine State Register, to 1. 2. Set HID0[EMCP] to 1. 00 Non-maskable interrupt 01 Critical interrupt 10 Machine check request 11 No NMI, critical Interrupt, or machine check request generated.
19 NWRE2	NMI Wakeup Request Enable 2. **NOTE:** If wakeup requests are disabled, the corresponding NDSS field must be set to 11 to disable wakeups from an NMI, critical interrupt, or machine check. 0 System wakeup requests from the corresponding NIF2 bit are disabled 1 A set NIF2 bit or set NOVF2 bit causes a system wakeup request
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 NREE2	NMI Rising-edge Events Enable 2. 0 Rising-edge event is disabled 1 Rising-edge event is enabled
22 NFEE2	NMI Falling-edge Events Enable 2. 0 Falling-edge event is disabled 1 Falling-edge event is enabled

Table continues on the next page...

WKPU_NCR field descriptions (continued)

Field	Description
23 Reserved	This field is reserved. Always write 0 to this field.
24 RLOCK	Reset Configuration Lock Register. Writing a 1 to this bit locks the configuration for the NMI until it is unlocked by a system reset . Writing a 0 has no effect.
25–26 RDSS	Reset Destination Source Select **NOTE:** If this field is configured for machine check exception, the following steps must be performed in order to receive the exception on the NMI Pad: 1. Set MSR[ME], Machine Check Enable field of the core's Machine State Register, to 1. 2. Set HID0[EMCP] to 1. 00 Reset request to RGM 01 No reaction 10 No reaction 11 No reaction
27 RWRE	Reset Wakeup Request Enable. **NOTE:** If this wake-up request is disabled, the corresponding RDSS field must be set to 11 to disable corresponding reset reaction. 0 System wakeup requests from the corresponding NSR[RIF] bit are disabled 1 A set NSR[RIF] bit or set NSR[ROVF] bit causes a system wakeup request
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 RREE	Reset Rising-edge Events Enable 0 Rising-edge event is disabled 1 Rising-edge event is enabled
30 RFEE	Reset Falling-edge Events Enable 0 Falling-edge event is disabled 1 Falling-edge event is enabled
31 Reserved	This field is reserved.

24.5.3 Wakeup/Interrupt Status Flag Register (WKPU_WISR)

This register holds the wakeup/interrupt flags.

NOTE

- This register is accessible only by 32-bit read/write operations.
- Status bits associated with on-chip wakeup sources are located to the left of the external wakeup/interrupt status bits and are read only. The wakeup for these sources must be configured and cleared at the on-chip wakeup source.

Also, the configuration registers for the external interrupts/ wakeups do not have corresponding bits.

Access: User read/write

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EIF																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WKPU_WISR field descriptions

Field	Description
0–31 EIF	External Wakeup/Interrupt Status Flag x. This flag can be cleared only by writing a 1. Writing a 0 has no effect. If enabled (IRER[x]), EIF[x] causes an interrupt request. 0 No event has occurred on the pad 1 An event as defined by WIREER and WIFEER has occurred

24.5.4 Interrupt Request Enable Register (WKPU_IRER)

This register is used to enable the interrupt messaging from the wakeup/interrupt pads to the interrupt controller.

NOTE

This register is accessible only by 32-bit read/write operations.

Access: User read/write

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EIRE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WKPU_IRER field descriptions

Field	Description
0–31 EIRE	External Interrupt Request Enable x. 0 Interrupt requests from the corresponding EIF[x] bit are disabled 1 A set EIF[x] bit causes an interrupt request

24.5.5 Wakeup Request Enable Register (WKPU_WRER)

This register is used to enable the system wakeup messaging from the wakeup/interrupt pads to the mode entry and power control modules.

NOTE

This register is accessible only by 32-bit read/write operations.

If a pin is disabled through this register, the corresponding bits in the WIFEER and WIREER registers must be written to 0 to ensure that the pin does not respond to any change.

Access: User read/write

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WRE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WKPU_WRER field descriptions

Field	Description
0–31 WRE	External Wakeup Request Enable x. 0 System wakeup requests from the corresponding EIF[x] bit are disabled 1 A set EIF[x] bit causes a system wakeup request

24.5.6 Wakeup/Interrupt Rising-Edge Event Enable Register (WKPU_WIREER)

This register is used to enable rising-edge triggered events on the corresponding wakeup/ interrupt pads.

NOTE

This register is accessible only by 32-bit read/write operations.

Access: User read/write

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IREE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WKPU_WIREER field descriptions

Field	Description
0–31 IREE	External Interrupt Rising-edge Events Enable x. 0 Rising-edge event is disabled 1 Rising-edge event is enabled

24.5.7 Wakeup/Interrupt Falling-Edge Event Enable Register (WKPU_WIFEER)

This register is used to enable falling-edge triggered events on the corresponding wakeup/interrupt pads.

NOTE

This register is accessible only by 32-bit read/write operations.

Access: User read/write

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IFEEx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WKPU_WIFEER field descriptions

Field	Description
0–31 IFEEx	External Interrupt Falling-edge Events Enable x. 1 Falling-edge event is enabled 0 Falling-edge event is disabled

24.5.8 Wakeup/Interrupt Filter Enable Register (WKPU_WIFER)

This register is used to enable an analog filter on the corresponding interrupt pads to filter out glitches on the inputs. The number of wakeups/interrupts supporting this feature is SoC dependent and can be configured to be between 1 and 32.

NOTE

This register is accessible only by 32-bit read/write operations.

Access: User read/write

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IFE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WKPU_WIFER field descriptions

Field	Description
0–31 IFE	External Interrupt Filter Enable x. Enable analog glitch filter on the external interrupt pad input. 0 Filter is disabled 1 Filter is enabled

24.5.9 Wakeup/Interrupt Pull Enable Register (WKPU_WIPER)

This register is used to enable a Pull on the corresponding interrupt pads to pull an unconnected wakeup/interrupt input to a voltage level (See the Chip-specific WKPU information for the exact level of voltage) .

NOTE

This register is accessible only by 32-bit read/write operations.

Access: User read/write

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPUE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

WKPU_WIPER field descriptions

Field	Description
0–31 IPUE	External Interrupt Pull Enable x. 0 Pull is disabled 1 Pull is enabled

24.6 Functional description

This section provides a functional description of the WKPU.

24.6.1 Non-maskable interrupts

The WKPU supports the generation of 3 types of interrupts per NMI input to the SoC. The WKPU supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event, though it creates an overrun condition.

Each NMI passes through a bypassable analog glitch filter.

NOTE

Glitch filter control and pad configuration should be done while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

24.6.1.1 NMI management

Each NMI can be enabled or disabled independently. This can be performed using the single NCR register laid out to contain all configuration bits for a given NMI in a single byte (see NMI Configuration Register (WKPU_NCR)). A pad defined as an NMI can be configured by the user to recognize interrupts with an active rising edge, an active falling edge or both edges being active. A setting of having both edge events disabled results in no interrupt being detected and should not be configured.

The active NMI edge is controlled by the user through the configuration of the NREE and NFEE bits.

Note

After reset, NREE and NFEE are set to '0', therefore the NMI functionality is disabled after reset and must be enabled explicitly by software.

Once a pad's NMI functionality has been enabled, the pad cannot be reconfigured in the IOMUX to override or disable the NMI. Please see chip specific section for details on NMI implementation.

The NMI destination interrupt is controlled by the user through the configuration of the NDSS bits. See NMI Configuration Register (WKPU_NCR) for details.

Each NMI supports a status flag and an overrun flag which are located in the NSR register (see NMI Status Flag Register (WKPU_NSR)). This register is a clear-by-write-1 register type, preventing inadvertent overwriting of other flags in the same register. The status flag is set whenever an NMI event is detected. The overrun flag is set whenever an NMI event is detected and the status flag is set (i.e. has not yet been cleared).

Note

The overrun flag is cleared by writing a '1' to the appropriate overrun bit in the NSR register. If the status bit is cleared and the overrun bit is still set, the pending interrupt will not be cleared.

During an NMI ISR, on wakeup of the SoC from an NMI, any writes to ECC-protected memory must have the correct ECC.

24.7 Initialization Information

This section discusses initialization for the following features:

- Glitch Filter and Pad Configuration
- Non-Maskable Interrupts
- Reset Request

24.7.1 Glitch Filter and Pad Configuration

Glitch filter control and pad configuration should be performed while the NMI is disabled in order to avoid erroneous triggering by glitches caused by the configuration process itself.

When enabling the glitch filter, do not enable the rising-/falling-edge events bits, i.e., the NREE, NFEE, RREE, and RFEE, bits in the same register write.

24.7.2 Non-Maskable Interrupts

When an NMI interrupt pin is first enabled, it is possible to get a false interrupt flag. To prevent a false interrupt request during pin interrupt initialization the user must do the following:

1. Mask interrupts by clearing NSR[NIF*n*] and NSR[NOVF*n*].
2. Clear NCR[NFEE*n*] and NCR[NREE*n*], preferably by writing 0 to all bits in the register.
3. Configure the analog glitch filter using NCR[NFE0].

4. Write appropriate bit values to the NCR register to configure:
    - NMI source (NDSS*n*).
    - Wakeup request (NWRE*n*).
    - Glitch filter (NFE0)
    - Rising/falling edge events enable (NFEE*n* and NREE*n*).
    - NMI configuration lock (NLOCK*n*).

24.7.3 Reset Request

When an NMI reset request pin is first enabled, it is possible to get a false reset request. To prevent a false reset request during pin reset request initialization, the user must do the following:

1. Mask reset requests by clearing NSR[RIF] and NSR[ROVF].
2. Clear NCR[RFEE] and NCR[RREE], preferably by writing 0 to all bits in this register.
3. Configure the analog glitch filter, as desired, using NCR[NFE0].
4. Write appropriate bit values to the NCR register to configure:
    - Reset destination source (RDSS]).
    - Reset wakeup request (RWRE).
    - Glitch filter (NFE0)
    - Rising/falling edge events enable (RFEE and RREE).
    - Reset configuration lock (RLOCK).

Chapter 25
PLL Digital Interface (PLLDIG)

25.1 Introduction

The PLL system is composed of a single analog PLL block and the digital interface (PLLDIG).

For low power operation, the chip provides an option to work with the PLL in either disabled or powered down modes.

25.2 Block Diagram

Please refer to the “Clocking chapter” of this *Reference Manual* to see the block diagram.

25.3 Features

The PLLDIG module has the following features:

- Reference clock pre-divider for finer frequency synthesis resolution.
- Reduced frequency divider for decreasing the PLL output clock frequency without causing the PLL to lose lock.
- Programmable frequency modulation.
- Lock detect circuitry reports when the PLL has achieved frequency lock, and continuously monitors lock status to report loss of lock conditions.
    - The loss of lock indication is sent to the FCCU via the system glue logic.
- Loss-of-clock detection for the reference and output clocks.

25.4 Modes of operation

PLLDIG operates in normal mode with a reference clock and PLL enabled.

NOTE

Mode changes are controlled by configuring the MC_ME Mode Configuration registers (MC_ME_<*mode*>_MC).

Normal mode is defined as the mode where the clocks are driven by the PLL. When using a crystal for the reference clock, reset should remain asserted until the oscillator has stabilized.

25.4.1 Normal mode with reference, PLL enabled

In normal mode, the PLL receives an input clock from a reference which can be prescaled by the pre-divider. The PLL multiplies the reference to create the output clock. The user must supply a crystal that is within the appropriate frequency range with the crystal manufacturers recommended external support circuitry, and a short signal route from the crystal to the MCU.

The PLL can generate either a frequency modulated clock or a non-modulated clock (for example, locked on a single frequency). The modulation rate, modulation depth, and modulation enable are programmed by configuring the PLLFM register.

25.5 Memory map and register definition

This section provides the memory map and detailed descriptions of all registers for configuring the PLL. The table below shows the memory map. Addresses are given as offsets from the module base address. All registers can be accessed using 8-bit, 16-bit or 32-bit addressing.

PLLDIG memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
18	PLL Calibration Register 3 (PLLDIG_PLLCAL3)	32	R/W	See section	25.5.1/585
20	PLLDIG PLL Control Register (PLLDIG_PLLCR)	32	R/W	See section	25.5.2/586

Table continues on the next page...

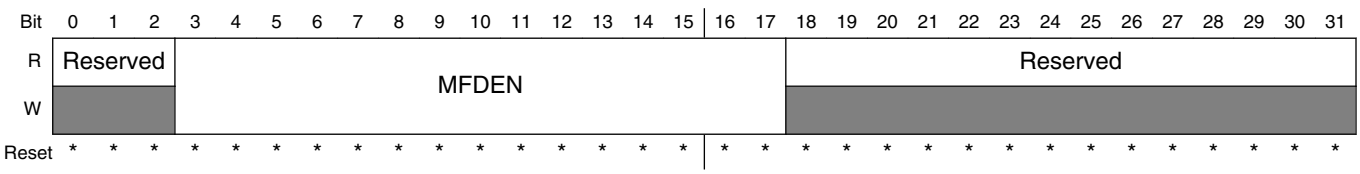
PLLDIG memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
24	PLLDIG PLL Status Register (PLLDIG_PLLSR)	32	R/W	See section	25.5.3/588
28	PLLDIG PLL Divider Register (PLLDIG_PLLDV)	32	R/W	See section	25.5.4/590
2C	PLLDIG PLL Frequency Modulation Register (PLLDIG_PLLFM)	32	R/W	See section	25.5.5/592
30	PLLDIG PLL Fractional Divide Register (PLLDIG_PLLFD)	32	R/W	See section	25.5.6/594

25.5.1 PLL Calibration Register 3 (PLLDIG_PLLCAL3)

PLLCAL3 contains the denominator of the fractional loop division factor.

Address: 0h base + 18h offset = 18h



* Notes:
- See "Clocking" chapter for reset value.

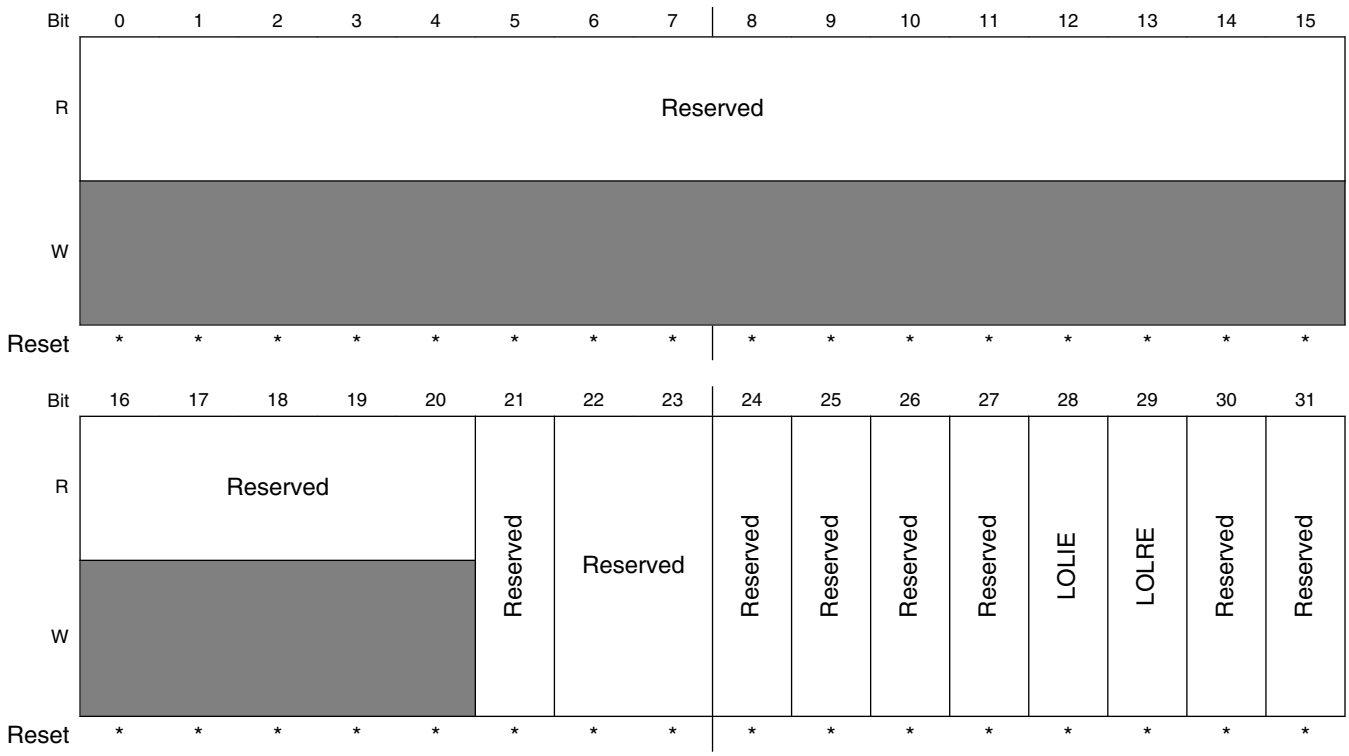
PLLDIG_PLLCAL3 field descriptions

Field	Description
0–2 Reserved	This field is reserved.
3–17 MFDEN	Denominator of fractional loop division factor. The value for this field must be greater than PLLFD[MFN] (see Clock configuration for details).
18–31 Reserved	This field is reserved.

25.5.2 PLLDIG PLL Control Register (PLLDIG_PLLCR)

The PLLCR is shown in the following table. It contains the operational state of the PLL, and is used to enable/disable PLL specific interrupts.

Address: 0h base + 20h offset = 20h



* Notes:
- See "Clocking" chapter for reset value.
- LOLIE field: Please refer to device "Clocking" chapter for details on configuring this field.

PLLDIG_PLLCR field descriptions

Field	Description
0–20 Reserved	This field is reserved.
21 Reserved	This field is reserved.
22–23 Reserved	This field is reserved.
24 Reserved	This field is reserved.
25 Reserved	This field is reserved.

Table continues on the next page...

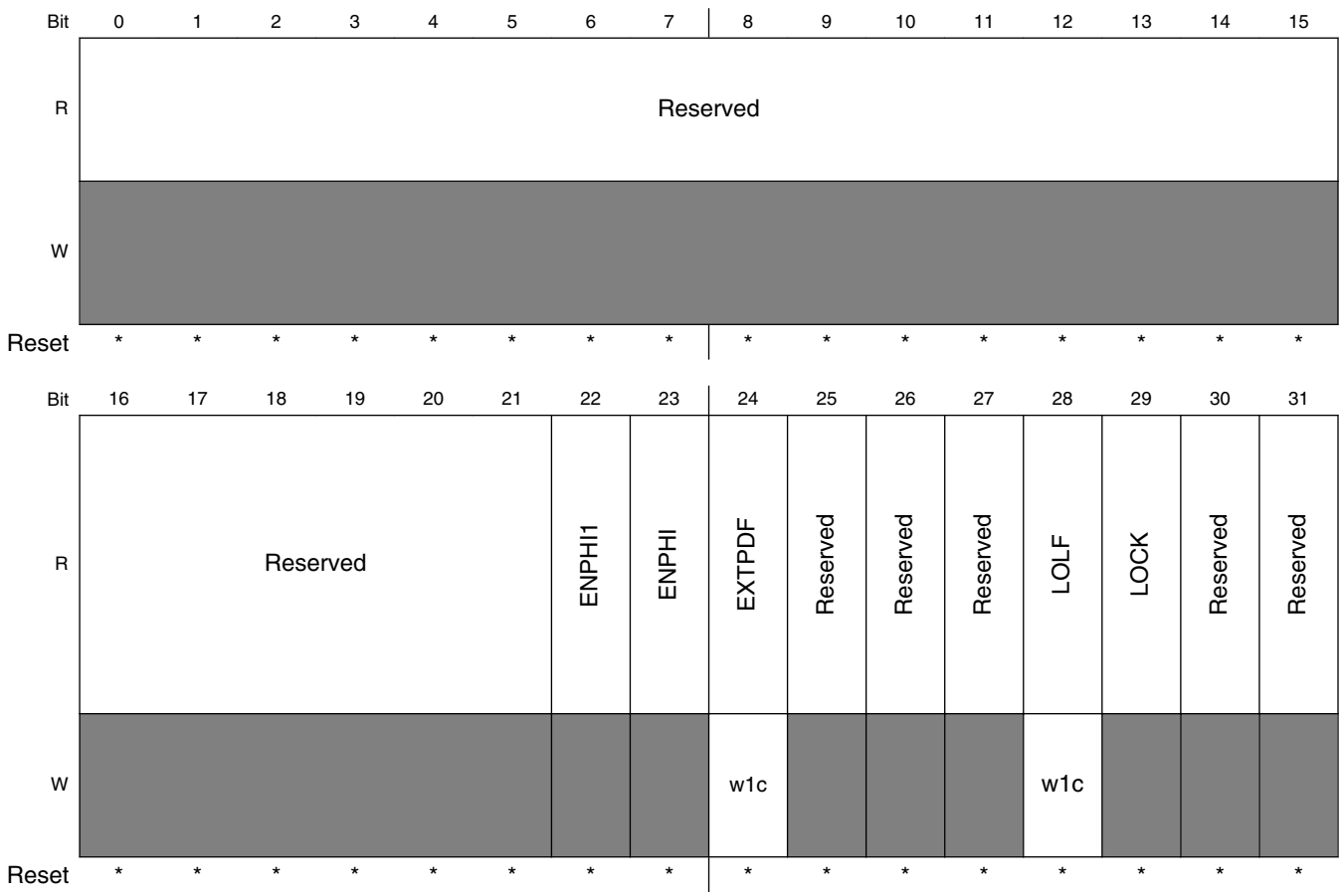
PLLDIG_PLLCR field descriptions (continued)

Field	Description
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLIE	Loss-of-lock interrupt enable. The LOLIE bit enables a loss-of-lock interrupt request when PLLSR[LOLF] = 1. If PLLSR[LOLF] = 0 or PLLCR[LOLIE] = 0, the interrupt request is ignored. The interrupt request only occurs in normal mode. 0 Ignore loss-of-lock. Interrupt not requested. 1 Enable interrupt request upon loss-of-lock.
29 LOLRE	Loss-of-lock reset enable. The PLLCR[LOLRE] bit determines whether system reset is asserted or not upon a loss-of-lock indication from PLL. If both PLLCR[LOLRE] = 1 and PLLCR[LOLIE] = 1, PLLCR[LOLRE] has precedence and the device asserts reset with the interrupt not being taken. 0 Ignore loss-of-lock. Reset not asserted. 1 Assert reset on loss-of-lock when operating in normal mode.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

25.5.3 PLLDIG PLL Status Register (PLLDIG_PLLSR)

PLLSR contains status flags showing the current state of the PLL.

Address: 0h base + 24h offset = 24h



* Notes:
- See "Clocking" chapter for reset value.

PLLDIG_PLLSR field descriptions

Field	Description
0–21 Reserved	This field is reserved.
22 ENPHI1	PLL PHI1 Output Enable 0 PLL_PHI1 output is disabled 1 PLL_PHI1 output is enabled
23 ENPHI	PLL PHI Output Enable

Table continues on the next page...

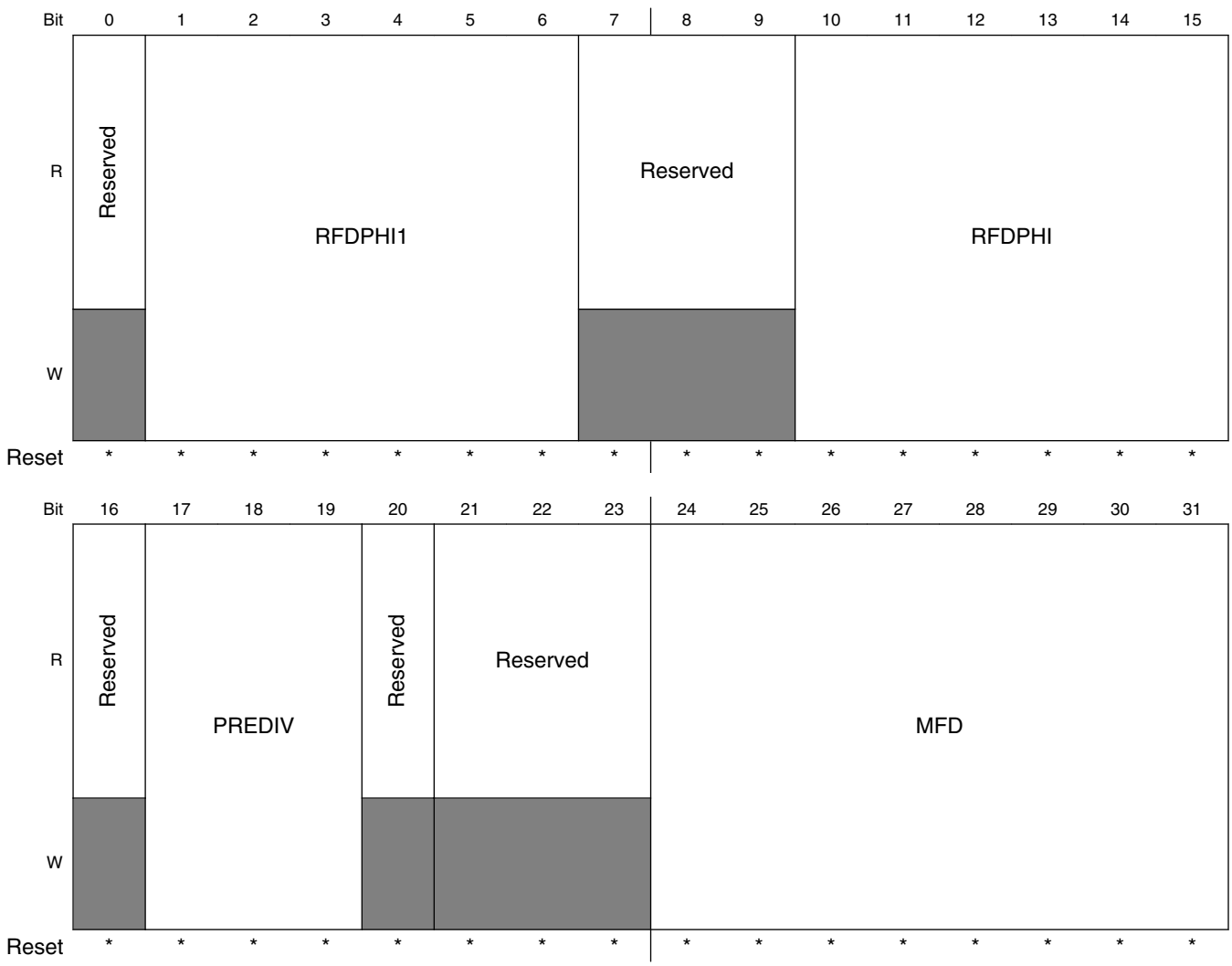
PLLDIG_PLLSR field descriptions (continued)

Field	Description
	0 PLL_PHI output is disabled 1 PLL_PHI output is enabled
24 EXTPDF	External Power Down Cycle Complete indication interrupt flag. This bit indicates that the external power down of the PLL is complete (for example, PLLs are powered down and then powered up). 0 PLLs not power cycled. 1 PLLs power down cycle is complete.
25 Reserved	This field is reserved.
26 Reserved	This field is reserved.
27 Reserved	This field is reserved.
28 LOLF	Loss-of-lock flag. This bit provides the interrupt request flag for the loss-of-lock condition. Software must write 1 to LOLF to clear it, writing 0 has no effect. This flag bit is sticky in the sense that if lock is reacquired, the bit will remain set until cleared by either writing 1 or asserting reset. Clearing this bit does not clear the interrupt request since this is handled by the MC_RGM. 0 No loss of lock detected. Interrupt service not requested. 1 Loss of lock detected. Interrupt service requested.
29 LOCK	Lock status bit. Indicates whether PLL has acquired lock. 0 PLL is unlocked. 1 PLL is locked.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

25.5.4 PLLDIG PLL Divider Register (PLLDIG_PLLDV)

The values of the reduced frequency divider (RFDPHI) and loop multiplication factor divider (MFD) can be modified at anytime, but the new values only become effective after the PLL is disabled, then reenabled.

Address: 0h base + 28h offset = 28h



* Notes:
- See "Clocking" chapter for reset value.

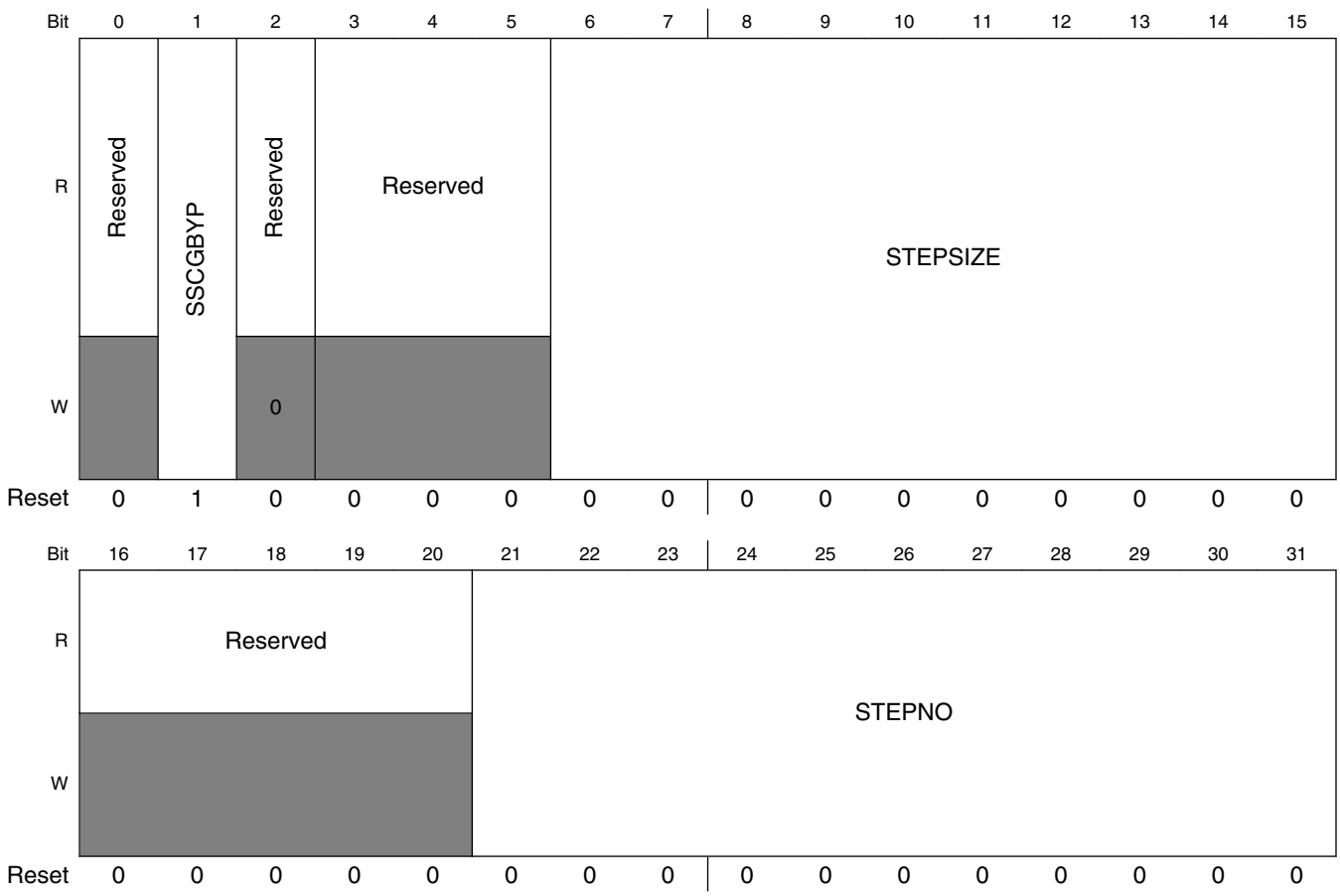
PLLDIG_PLLDV field descriptions

Field	Description
0 Reserved	This field is reserved.
1–6 RFDPHI1	PHI1 reduced frequency divider. This field determines the VCO clock post divider for driving the PHI1 output clock. The settings for RFDPHI1 are: 000000 Divide by 2 000001 Divide by 4 000010 Divide by 8 000011 Divide by 16 0001xx Divide by 32 All other values are reserved.
7–9 Reserved	This field is reserved.
10–15 RFDPHI	PHI reduced frequency divider. This 6-bit field determines the VCO clock post divider for driving the PHI output clock. The settings for RFDPHI are as follows: 000000 Divide by 2 000001 Divide by 4 000010 Divide by 8 000011 Divide by 16 0001xx Divide by 32 All other values are reserved.
16 Reserved	This field is reserved.
17–19 PREDIV	Input clock predivider. This field controls the value of the divider on the input clock. The output of the predivider circuit generates the reference clock to the PLL analog loop. 000 Divide by 1 001 Divide by 1 010 Divide by 2 011 Divide by 3 100 Divide by 4 101 Divide by 5 110 Divide by 6 111 Reserved
20 Reserved	This field is reserved.
21–23 Reserved	This field is reserved.
24–31 MFD	Loop multiplication factor divider. This field controls the value of the divider in the feedback loop. The value specified by the MFD bits establishes the multiplication factor applied to the reference frequency. Divider value = MFD, where MFD has the range 10...150 (Ah...96h). All other values are reserved.

25.5.5 PLLDIG PLL Frequency Modulation Register (PLLDIG_PLLFM)

The PLLFM register controls enables frequency modulation, and provides controls for modulation mode selection, modulation depth and modulation rate. The PLL would then be required to be power cycled to regain normal functionality.

Address: 0h base + 2Ch offset = 2Ch



PLLDIG_PLLFM field descriptions

Field	Description
0 Reserved	This field is reserved.
1 SSCGBYP	Modulation enable. This bit enables spectrum modulation. 0 Spread spectrum modulation is not bypassed. 1 Spread spectrum modulation is bypassed.

Table continues on the next page...

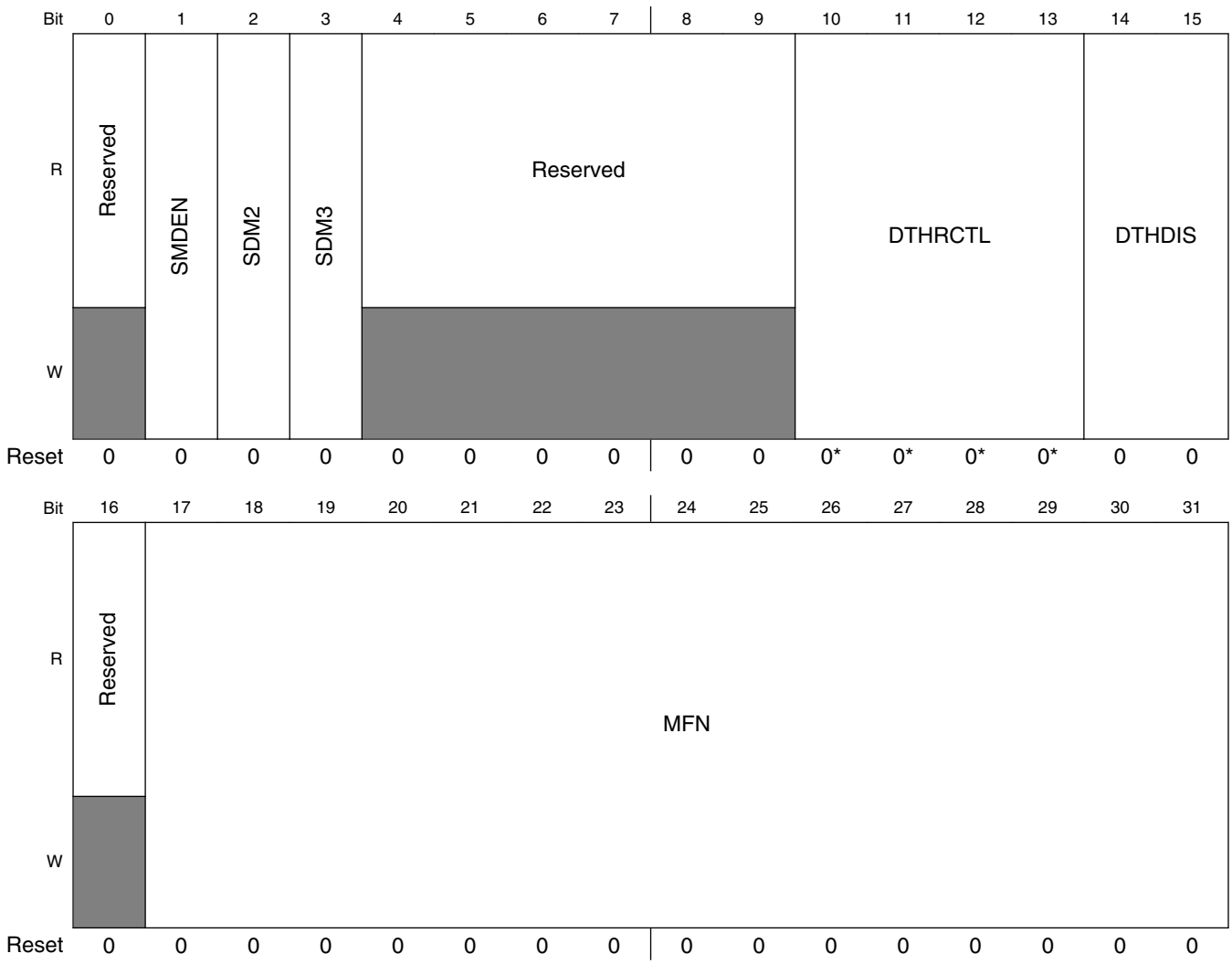
PLLDIG_PLLFM field descriptions (continued)

Field	Description
2 Reserved	This field is reserved.
3–5 Reserved	This field is reserved.
6–15 STEPSIZE	Modulation period. STEPSIZE is the modulation period variable derived from the formulas shown in Frequency modulation.
16–20 Reserved	This field is reserved.
21–31 STEPNO	Increment step. This field is the STEPNO variable derived from the formulas shown in Frequency modulation.

25.5.6 PLLDIG PLL Fractional Divide Register (PLLDIG_PLLFD)

The PLLFD register provides the enable and fractional divide factor for the loop divider. The PLL would then require power cycling to regain normal functionality.

Address: 0h base + 30h offset = 30h



* Notes:
- DTHRCTL field: See field description for details.

PLLDIG_PLLFD field descriptions

Field	Description
0 Reserved	This field is reserved.

Table continues on the next page...

PLLDIG_PLLFD field descriptions (continued)

Field	Description
1 SMDEN	Sigma Delta Modulation Enable 0 Sigma delta modulation disabled 1 Sigma delta modulation enabled
2 SDM2	Second Order Sigma Delta Modulation Select See Table 25-1 for proper SDM2 and SDM3 field configuration. 0 2nd order sigma delta modulation disabled. 1 2nd order sigma delta modulation enabled.
3 SDM3	Third Order Sigma Delta Modulation Select Table 25-1 shows how to configure the SDM3 and SDM2 fields for proper operation. **Table 25-1. SDM2/SDM3 field configuration** SDM2 | SDM3 | Order of modulation 0 | 0 | Not applicable 0 | 1 | Not applicable 1 | 0 | 2nd order sigma delta modulation 1 | 1 | 3rd order sigma delta modlation **NOTE:** For SSCG mode, PLLDIG must be configured for 3rd order sigma delta modulation. 0 3rd order sigma delta modulation disabled. 1 3rd order sigma delta modulation enabled.
4–9 Reserved	This field is reserved.
10–13 DTHRCTL	Dither Control. **NOTE:** DTHRCTL may be written, but for normal operation this field must be 0000b. All other values are reserved.
14–15 DTHDIS	Dither Disable. 00 Dither enabled 01 Dither disabled 10 Dither enabled 11 Dither disabled
16 Reserved	This field is reserved.
17–31 MFN	Numerator for fractional loop division factor (see Clock configuration).

25.6 Functional description

This section explains the operation and configuration of the PLLDIG module.

25.6.1 Input clock frequency

The PLL is designed to operate over an input clock frequency range. The operating ranges for the PLL is discussed in detail in the *Data Sheet*.

25.6.2 Clock configuration

The relationship between input and output frequency is determined by programming the PLLDV, PLLCAL3, and PLLFD registers, and calculated according to the following equation:

$$f_{\text{pll_phi}} = f_{\text{pll_ref}} \times \left(\frac{\text{PLLDV[MFD]} + \frac{\text{PLLFD[MFN]}}{(\text{PLLCAL3[MFDEN]} + 1)}}{\text{PLLDV[PREDIV]} \times 2 \times \left(2^{\text{PLLDV[RFDPHI]}}\right)} \right)$$

Equation 1. PLL PHI Output Frequency

The relationship between the VCO frequency (f_{VCO}) and the output frequency of the PLL is determined by the PLLDV, PLLCAL3 and PLLFD registers, according to the following equation:

$$f_{\text{pll_VCO}} = \frac{f_{\text{pll_ref}}}{\text{PLLDV[PREDIV]}} \times \left(\text{PLLDV[MFD]} + \frac{\text{PLLFD[MFN]}}{(\text{PLLCAL3[MFDEN]} + 1)} \right)$$

Equation 2. PLL VCO Frequency

When programming the PLL, user software must not violate the maximum system clock frequency or max/min VCO frequency specification of the PLL.

The lock signal and PLLSR[LOCK] flag are immediately negated if the fields of the PLLDV register are changed without powering down the analog PLL.

When any of these events occur, an internal timer is initialized to count 64 cycles of the PLL input clock. During this period (64 cycles and a few extra clock cycles for synchronization, for example, 64 to 72 cycles), the PLLSR[LOCK] flag is held negated. After the timer expires, the PLLSR[LOCK] flag reflects the value coming from the PLL lock detection circuitry. To prevent an immediate reset, the PLLCR[LOLRE] bit of the PLL must be cleared before doing any of the above operations.

Note

The PLL must be powered down and powered up by configuring, then executing, a mode change in the MC_ME_<*mode*>_MC before PLLDV[PREDIV], PLLDV[MFD], or the input clocks are modified.

The recommended procedure to program the PLL and engage normal mode is shown in Initialization information.

25.6.3 Fault Collection and Control Unit (FCCU)

When a loss of lock condition is detected, a signal is sent to the INTC as well as the FCCU. The FCCU then uses this signal to determine an appropriate action.

25.6.4 Frequency modulation

Frequency modulation uses a triangular profile as shown in Figure 25-1. The modulation frequency and depth are controlled using PLLFM[STEPSIZE] and PLLFM[STEPNO].

NOTE

Frequency Modulation is only possible if,

$$\text{PLLFM[STEPSIZE]} \times \text{PLLFM[STEPNO]} < (\text{PLLCAL3[MFDEN]} + 1).$$

PLLFM[SSCGBYP] must be cleared and PLLFD[SMDEN] must be set for the frequency modulation mechanism to be enabled.

Maximum peak-to-peak Modulation Depth (MD) = (PLL reference clock ÷ VCO frequency) × 100.

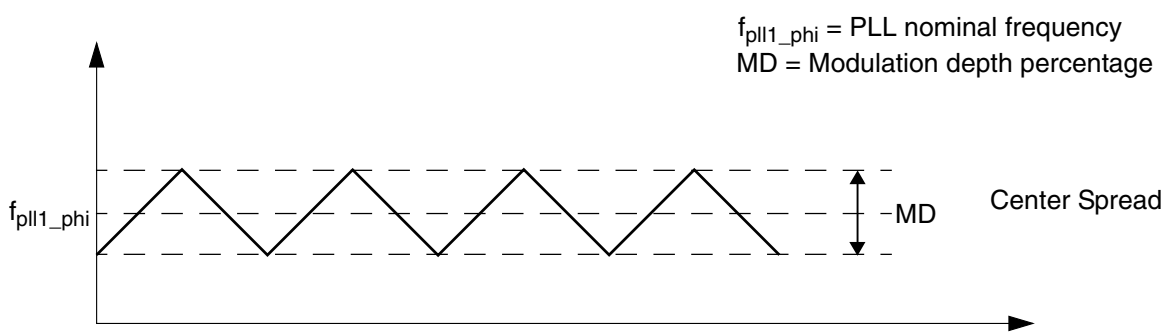


Figure 25-1. Triangular frequency modulation

NOTE

The device maximum operating frequency includes the frequency modulation. If center modulation is used, the f_{sys} must be below f_{max} by MD percentage such that $f_{max} = f_{sys} \times (1 + (MD\% / 100))$.

The following equations define how to calculate PLLFM[STEPSIZE] and PLLFM[STEPNO] based on the output frequency of the feedback divider (f_{ref}), the modulation frequency (f_{mod}) and the modulation depth percentage (MD).

$$PLLFM[STEPSIZE] = \text{round}\left(\frac{MD\% \times N \times PLLCAL3[MFDEN] + 1}{PLLFM[STEPNO] \times 100}\right)$$

Equation 3. PLLFM[STEPSIZE] Calculation

Where MD is the % Modulation Depth, and N is shown in Equation 4 on page 598.

$$N = PLLDV[MFD] + \frac{PLLFD[MFN]}{PLLCAL3[MFDEN] + 1}$$

Equation 4. Calculation of N

The equation to determine PLLFM[STEPNO] is shown in Equation 5 on page 598.

$$PLLFM[STEPNO] = \text{round}\left(\frac{f_{ref}}{PLLDV[PREDIV] \times 2 \times f_{mod}}\right)$$

Equation 5. PLLFM[STEPNO] Calculation

25.6.5 PLL Wakeup from low power mode

When the PLL is running normally and locked, the PLL operating point is sampled and stored.

On entering low power state, register configurations will not be maintained. However, the PLL register configurations are saved. On wake-up from a low power mode, the stored values of the PLL are written back to the registers. This process allows the PLL to acquire a faster lock.

25.6.6 Calibration feature

The PLL has a mechanism to calibrate the VCO. After the PLL exits reset, the PLL undergoes an automatic tuning of the VCO. The automatic tuning of the VCO will assist the VCO to oscillate close to the locking frequency. During the automatic tuning the main loop of PLL will not be working. Upon completion of calibration, the main loop of the PLL is closed which fine tunes the VCO to lock at the desired frequency.

25.7 Initialization information

Coming out of reset the PLL is disabled per the DRUN mode configuration register, MC_ME_DRUN_MC. The PLLDIG initialization procedure is described in the "Clocking" chapter.

NOTE

See the "Mode Entry Module (MC_ME)" chapter in this *Reference Manual* more details.

Chapter 26
Slow Internal 128 kHz Oscillator (SIRC)

26.1 Introduction

This internal RC oscillator provides the 128 kHz clock with high accuracy, low power consumption, fast stabilization, and fast recovery.

26.2 Features

Key features of the 128 kHz SIRC module are as follows:

- Operating voltage generated internally from V_{DD_LV}
- Nominal frequency (F_{nom}) of 128 kHz
- Frequency accuracy
- Low current consumption

26.3 Memory map and register definition

The following section shows the memory map and explains the register.

SIRC memory map

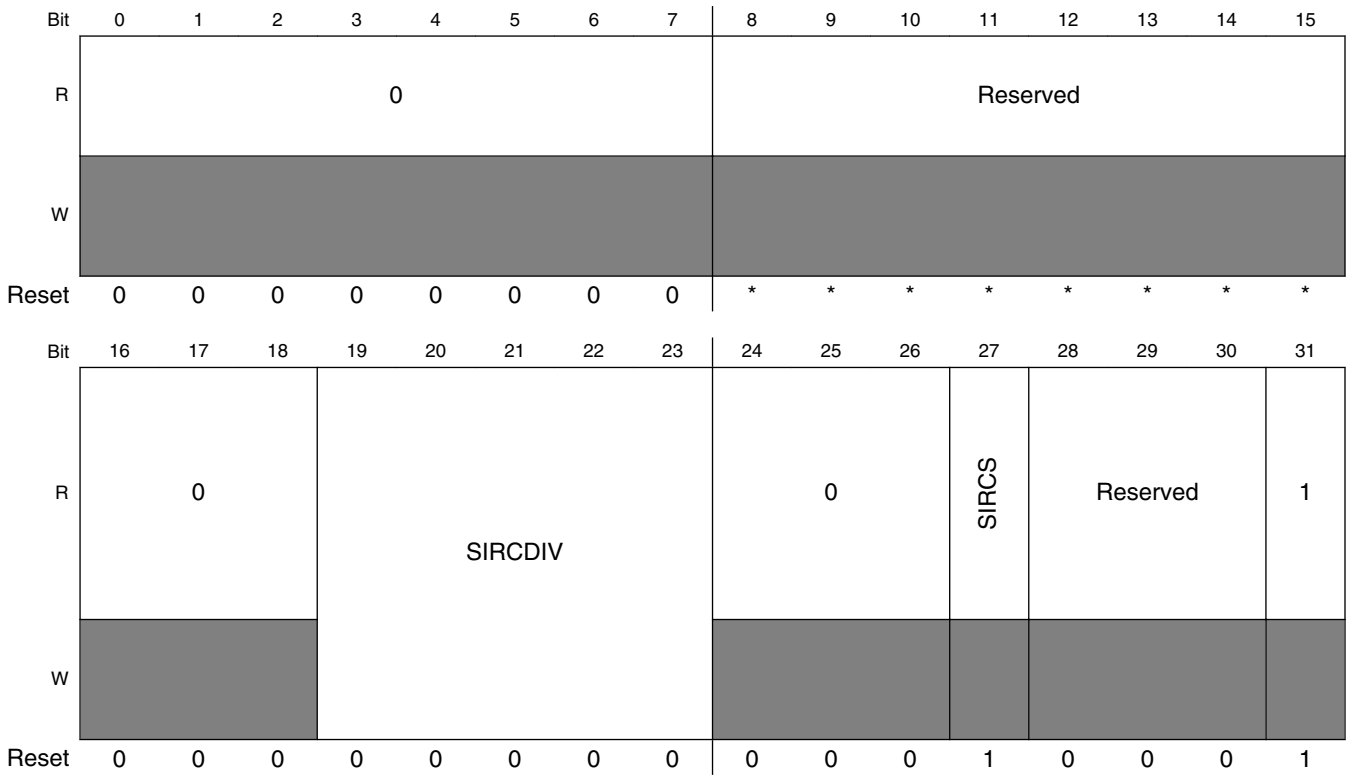
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control register (SIRC_CTL)	32	R/W	See section	26.3.1/601

26.3.1 Control register (SIRC_CTL)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 0h offset = 0h



* Notes:

- Reserved field: These fields may be written during reset, and their values will be change. Please consult the "Clocking Overview" chapter in this document for more information on field values after reset.

SIRC_CTL field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 Reserved	This field is reserved.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 SIRCDIV	The SIRC Divider bits select the clock divider . factor 00000 1 00001 2 00010 3 11111 32
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

SIRC_CTL field descriptions (continued)

Field	Description
27 SIRCS	Describes the SIRC clock status. 0 IRC is not providing a stable clock. 1 IRC is providing a stable clock.
28–30 Reserved	This field is reserved.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

26.4 Functional description

The clock provided by SIRC is always on in the normal mode after startup. The stabilization time from startup is shorter than 16 clock cycles. The SIRC_CTL[SIRCS] bit is set once the clock is stable.

The clock can be further divided by a configurable division factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by SIRC_CTL[SIRCDIV]. The on-off in standby mode is configurable by writing to ME_STANDBY0_MC[SIRCON].

Chapter 27
Fast Internal 16 MHz RC Oscillator (FIRC)

27.1 Introduction

This internal RC oscillator provides the 16 MHz clock with high accuracy, low-power, fast stabilization, and fast recovery.

27.2 Features

Key features of the 16 MHz FIRC module include:

- Operation from internally generated power supply (3 V or 5 V, with support down to 2.7 V)
- Nominal frequency (F_{nom}) of 16 MHz
- Frequency accuracy
- Low current consumption
- Ultra fast stabilization, enabling fast recovery in less than 16 cycles

27.3 Memory map and register definition

The following section shows the memory map and explains the register.

FIRC memory map

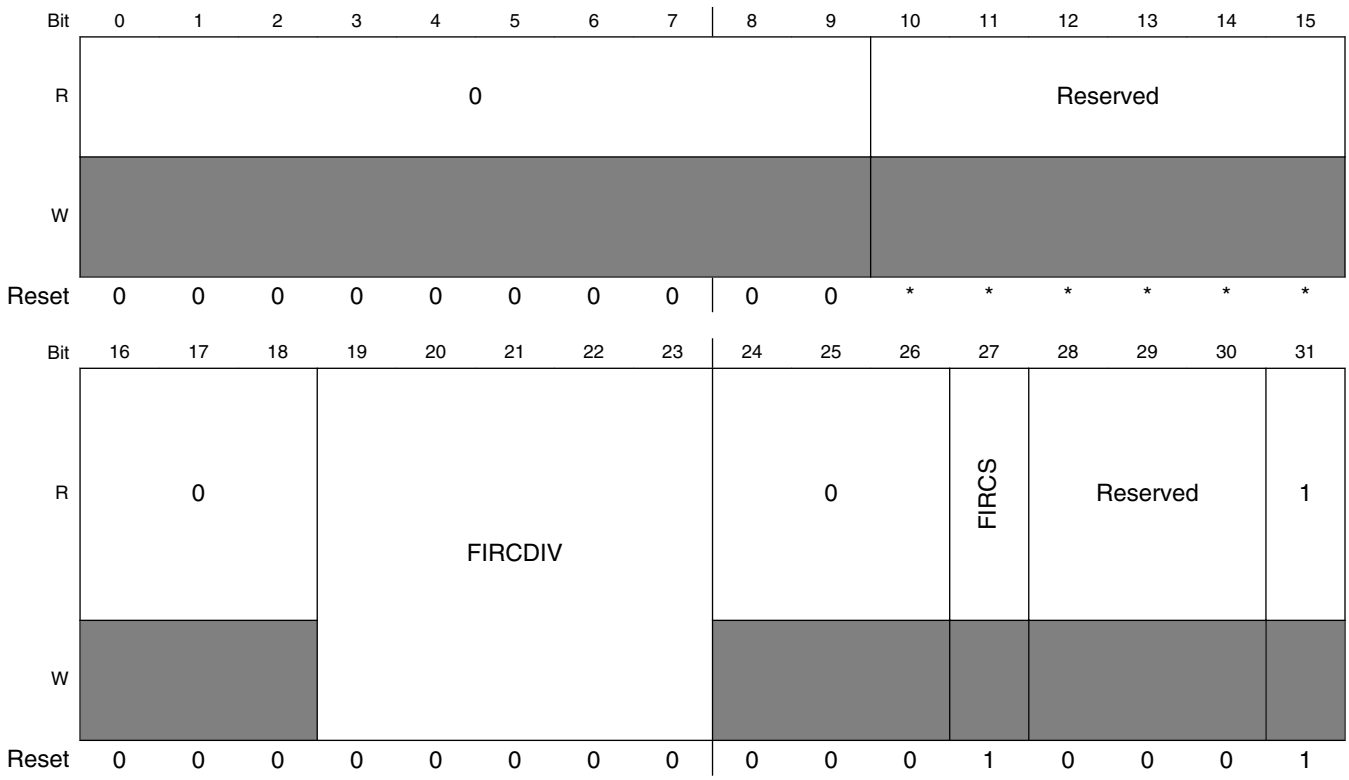
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control register (FIRC_CTL)	32	R/W	See section	27.3.1/606

27.3.1 Control register (FIRC_CTL)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 0h offset = 0h



* Notes:

- Reserved field: These reserved fields may be written during reset, and their values will be change. Please consult the "Clocking Overview" chapter in this document for more information on field values after reset.

FIRC_CTL field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 Reserved	This field is reserved.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 FIRCDIV	The FIRC Divider bits select the clock divider factor. 00000 1

Table continues on the next page...

FIRC_CTL field descriptions (continued)

Field	Description
	00001 2 00010 3 11111 32
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 FIRCS	Describes the FIRC clock status. 0 FIRC is not providing a stable clock. 1 FIRC is providing a stable clock.
28–30 Reserved	This field is reserved.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.

27.4 Functional description

The clock provided by FIRC is always on in the normal mode after startup. The stabilization time from startup is less than 16 clock cycles. Once the clock is stable, the FIRC_CTL[FIRCS] bit is set.

The clock can be further divided by a configurable division factor in the range 1 to 32 to generate the divided clock to match system requirements. This division factor is specified by FIRC_CTL[FIRCDIV]. The on-off in Standby mode is configurable by writing to ME_STANDBY0_MC[FIRCON].

Chapter 28
Slow External 32 KHz Oscillator (SXOSC)

28.1 Introduction

The slow external 32 KHz oscillator (SXOSC) provides a 32 KHz clock with control and status registers accessible for use by applications.

28.2 Features

Key features of the 32 KHz SXOSC module include:

- Operation from internally generated power supply (3 V or 5 V)
- Frequency (F_{nom}) — 32 KHz crystal
- Voltage and Frequency filtering to guarantee clock frequency and stability
- Power Consumption target: 5 µA
- Startup time lower than 2 sec

28.3 Memory map and register definition

The following section shows the memory map and explains the register.

SXOSC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SXOSC_CTL	32	R/W	See section	28.3.1/610

28.3.1 SXOSC_CTL

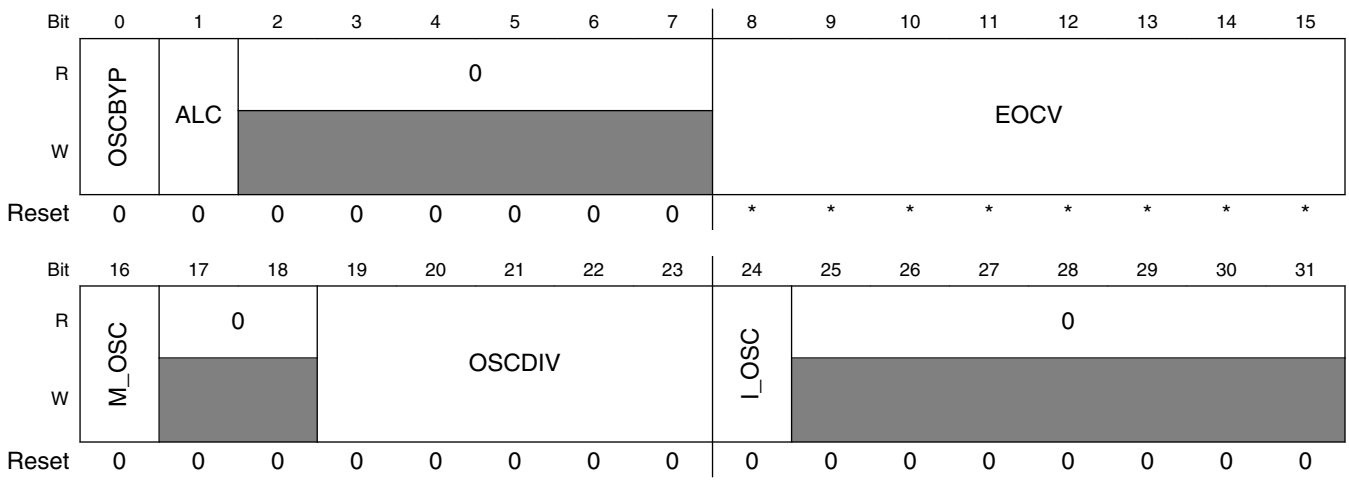
Access:

- Supervisor read/write
- User read/write

NOTE

"Write should be restricted to SXOSC when not being used as a clock".

Address: 0h base + 0h offset = 0h



* Notes:
- EOCV field: See Clocking chapter for reset value.

SXOSC_CTL field descriptions

Field	Description
0 OSCBYP	Oscillator start-up delay bypass This bit specifies whether to bypass the EOCV counter, which provides startup stabilization time for the crystal oscillator. When this bit is set, the oscillator starts immediately when the SXOSC is selected, instead of waiting for the stabilization time to elapse. The user can only set this bit. System reset is needed to clear this bit. 0 The EOCV counter is not bypassed, and the oscillator starts after the stabilization period has elapsed.. 1 The EOCV counter is bypassed, and the oscillator starts immediately.
1 ALC	Auto Level Control The ALC bit controls the current provided to the crystal. When set, the amplitude of the crystal is reduced by approximately one-third to one-half. This helps limit noise, limits power consumption, and may improve crystal life. When cleared, the amplitude of the crystal is at its maximum, which may help to prevent erratic clock generation in a very noisy environment.The customer should initially set this bit to see whether their system will maintain good clock generation at the lower power level.

Table continues on the next page...

SXOSC_CTL field descriptions (continued)

Field	Description
	0 Auto level control is disabled. 1 Auto level control is enabled.
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 EOCV	End of Counter Value These bits specify the end of the count value to be used for comparison by the SXOSC stabilization counter OSCCNT after reset or whenever it is switched on from the off state. This counting period ensures that the external oscillator clock signal is stable before it can be selected by the system. When the oscillator counter reaches the value EOCV*512, the oscillator available interrupt request is generated. The reset vaule of this field depends on the device specification. The OSCCNT counter is kept under reset if oscillator bypass mode is selected.
16 M_OSC	SXOSC clock interrupt mask 0 SXOSC clock interrupt is masked. 1 SXOSC clock interrupt is enabled.
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 OSCDIV	The SXOSC Divider bits select the clock divider factor. 00000 1 00001 2 00010 3 .. --- 11111 32
24 I_OSC	SXOSC clock interrupt This bit is set by hardware when the OSCCNT counter reaches the count value of EOCV × 512. It is cleared by software. 0 No SXOSC interrupt has occurred. 1 SXOSC interrupt is pending.
25–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

28.4 Functional description

The SXOSC frequency oscillator includes an internal oscillator driver, a controller, and external crystal circuitry. It provides an output clock that can be used to clock a PLL or as a reference clock to specific modules, depending on system needs. The SXOSC can be powered down and then restarted by the user whenever required.

Whenever the SXOSC is switched on, the OSCCNT counter starts. When it reaches the value of EOCV × 512, the SXOSC clock is made available to the system. The availability status is provided. Depending on the setting of the SXOSC_CTL[I_OSC] bit, an interrupt will be generated if the SXOSC_CTL[M_OSC] is set.

The SXOSC can be bypassed by setting the SXOSC_CTL[OSCBYP] bit. This can only be set by the user. It is disabled in default. In this bypass mode (OSCBYP is set), the output clock has the same polarity as the external clock applied on the EXTAL pin. The oscillator status is forced to 1, and the bypass configuration is independent of the power-down mode of the oscillator.

The SXOSC output can be further divided by a configurable factor in the range 1 to 32 to generated the divided clock to match system requirements. This factor is specified by SXOSC_CTL[OSCDIV].

Chapter 29
Fast OSC Digital Interface (FXOSC)

29.1 Introduction

The FXOSC digital interface is used to control the on-chip oscillator (FXOSC) and provide the register interface for the programmable features. Selection of the FXOSC as a source clock is controlled by the Clock Generation Module (MC_CGM). The FXOSC is enabled and disabled by configuring the Mode Entry Module (MC_ME).

The programmable features of the FXOSC digital interface are as follows:

- Oscillator power-down control and status
- Oscillator startup delay
- Oscillator clock available interrupt
- Oscillator bypass mode control
- Output clock division factors ranging from 1, 2, 3…32
- Crystal oscillator mode selection
    - Loop Controlled Pierce Mode (LCP)
    - Full Swing Pierce Mode (FSP)

29.2 Functional description

The FXOSC digital interface provides control of the on-chip oscillator and the register interface for the mode, internal divider, and startup delay control. The oscillator provides an output clock that is used as an input to the PLL and can be selected as the reference clock for many of the peripherals on the device.

The table below shows the possible configurations of the FXOSC.

Table 29-1. FXOSC configurations

OSC EN (from MC_ME)	FXOSC_CTL[OSCBYP]	XTAL	EXTAL	FXOSC Output	FXOSC MODE
0	0	Crystal, or left opened	Crystal, or left opened	0	Power down, IDDQ
0	1	x	external clock	Undefined	Reserved, OSC Disabled
1	1	x	external clock	XTALOUT	Bypass, OSC Enabled
1	0	crystal	crystal	XTALOUT	Normal, OSC Enabled

29.2.1 Oscillator power-down control and status

Enabling and disabling of the oscillator is done in the Mode Entry (MC_ME) module (see "Mode Entry Module (MC_ME)" chapter). The ME_<*mode*>_MC[XOSCON] controls the power down of oscillator based on the current device mode (see "MC_ME Memory Map" table for available *modes*). The status of the FXOSC is shown in the MC_ME_GS[S_XOSC].

29.2.2 Oscillator startup delay

In order to allow the oscillator to reach full amplitude before use, the internal counter (OSCCNT) is started when the device exits reset. The counter is driven by the oscillator output clock. When the counter reaches FXOSC_CTL[EOCV] × 512, a signal is sent to the MC_ME module, allowing a mode switch to the oscillator (when used as a system clock or an input to the PLL).

The default value for this field after reset depends on implementation and can be found in the FXOSC Device Configuration. After reset, software can clear or change the EOCV field in order to shorten or lengthen the delay until the oscillator is ready.

Note

It is advised that if the EOCV value is modified it should be done as early as possible in user code. This will ensure that the new value will be used before OSCCNT reaches the count value.

29.2.3 Oscillator clock available interrupt

An interrupt can be generated when the count value is reached (OSCCNT = FXOSC_CTL[EOCV] × 512). The FXOSC_CTL[I_OSC] will be set and an interrupt will be generated if the interrupt mask is set (FXOSC_CTL[M_OSC] = 1).

29.2.4 Oscillator bypass mode

The oscillator circuit can be bypassed by setting FXOSC_CTL[OSCBYP]. This field can only be set by software and is only cleared by a system reset. In bypass mode, the oscillator is disabled and the input clock on the EXTAL pins is level-shifted and driven to the logic on the device. The power-down mode has priority over the bypass configuration for FXOSC.

NOTE

The external oscillator must be running before the FXOSC is turned off. If FXOSC is being turned on, the status will be updated only after the clock starts toggling.

NOTE

The FXOSC will automatically be enabled until RTC auto calibration is active. Once RTC auto calibration is complete, the FXOSC will transition to power-down.

NOTE

Power down from the MC_ME will have no effect on the oscillator. The bypass clock will continue running in the device. It is ensured that mode transitions from the MC_ME complete successfully by providing bypass information to the MC_ME.

29.2.5 Crystal oscillator mode selection

There are two oscillator mode selections possible, Full Swing Pierce mode and Loop Controlled Pierce mode. The operating mode is selected by configuring the FXOSC_CTL[OSCM].

29.2.5.1 Loop Controlled Pierce mode

Loop Controlled Pierce mode (LCP) is the default oscillator mode. LCP has the following attributes:

- Provides a smaller frequency range
- Limits the amplitude of the crystal oscillation utilizing Amplitude Loop Control (ALC)
- Reduced EMI
- Reduced power consumption
- Shorter settling time
- Crystal is exposed to reduced stress since the amplitude is reduced
- Reduced number of external components, feedback resistor already integrated

29.2.5.2 Full Swing Pierce mode

Full Swing Pierce mode (FSP) has the following attributes when compared with LCP:

- Provides a wider frequency range
- Less susceptible to noise
- The crystal amplitude is rail-to-rail
  - Distorted oscilation with several harmonics
  - Increased EMI
- Slower settling time
- Feedback resistor and 300 ohms of current limitation are already integrated.

NOTE

Consult with crystal supplier to determine the need of extra resistance in XTAL to avoid crystal overdrive.

29.2.6 Oscillator divider

The oscillator output can be divided down if needed by the user. The CTL[OSCDIV] allows for the output of the FXOSC to be divided by values from 1…32 in increments of 1.

29.3 Memory map and register definition

FXOSC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	FXOSC Control Register (FXOSC_CTL)	32	R/W	See section	29.3.1/617

29.3.1 FXOSC Control Register (FXOSC_CTL)

NOTE

After reset, writing the FXOSC_CTL register would override the fuse values with the IPS write or reset value (depending upon the type of register access).

NOTE

The FXOSC_CTL is only writable in supervisor mode.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OSCBYP	OSCM	Reserved	Reserved					EOCV							
W																
Reset	*	*	0	0	0	0	0	0	*	*	*	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	M_OSC	0	0	OSCDIV					I_OSC	0			0	0	0	0
W									w1c							
Reset	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:
- I_OSC field: See Clocking chapter for reset value.
- M_OSC field: See Clocking chapter for reset value.

- EOCV field: See Clocking chapter for reset value.
- OSCM field: See Clocking chapter for reset value.
- OSCBYP field: See Clocking chapter for reset value.

FXOSC_CTL field descriptions

Field	Description
0 OSCBYP	Crystal Oscillator bypass This bit specifies whether the oscillator should be bypassed or not. Software can only set this bit. System reset is needed to reset this bit. 0 Oscillator output is used as root clock. 1 EXTAL is used as root clock.
1 OSCM	Crystal Oscillator Mode. This bit selects the oscillator mode when not in bypass. **NOTE:** User should not change the OSCM bit while crystal is being used. 0 FSP mode (full swing pierce mode) 1 LCP mode (loop controlled pierce mode)
2 Reserved	This field is reserved.
3–7 Reserved	This field is reserved.
8–15 EOCV	End of Count Value These bits specify the end of count value to be used for comparison by the oscillator stabilization counter OSCCNT after reset or whenever it is switched on from the off state. This counting period ensures that external oscillator clock signal is stable before it can be selected by the system. When oscillator counter reaches the value EOCV × 512, oscillator available interrupt request is generated. The OSCCNT counter is kept under reset if oscillator bypass mode is selected. **NOTE:** In order to find the appropriate EOCV value, it should be ensured that the internal counter be run for at least the stabilization time of the crystal as given in the Data Sheet.
16 M_OSC	Crystal oscillator clock interrupt mask This bit masks the I_OSC interrupt bit. 0 Crystal oscillator clock interrupt is masked. 1 Crystal oscillator clock interrupt is enabled.
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 OSCDIV	Crystal oscillator clock division factor These bits specify the crystal oscillator output clock division factor. The output clock is divided by the factor OSCDIV+1.
24 I_OSC	Crystal oscillator clock interrupt This bit is set by hardware when OSCCNT counter reaches the count value EOCV × 512. It is cleared by software by writing 1.

Table continues on the next page...

FXOSC_CTL field descriptions (continued)

Field	Description
	0 No oscillator clock interrupt occurred. 1 Oscillator clock interrupt pending.
25–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Chapter 30
Clock Monitor Unit (CMU)

30.1 Introduction

The Clock Monitor Unit (CMU), also referred to as Clock Quality Checker or Clock Fault Detector, serves three purposes:

- Measures the frequency of clock sources CLKMT0_RMN/CLKMT1/CLKMT2 with CLKMN0_RMT as the reference clock
- Monitors CLKMN0_RMT frequency with CLKMT0_RMN as reference clock
- Monitors CLKMN1 frequency with CLKMT0_RMN as reference clock and detects if the monitored clock frequency leaves an upper or lower frequency boundary

NOTE

See the "Clocking" chapter for chip-specific sources used by the CMU.

One of the tasks is to supervise the integrity of the various clock sources on the chip, for example CLKMN0_RMT or CLKMN1. If the monitored clock frequency is less than the reference clock, or it violates an upper or lower frequency boundary, the CMU detects and reports this event. These events signal the FCCU, which can take the necessary corrective actions as its configuration dictates.

The CMU can monitor CLKMN0_RMT, which must have a frequency higher than that of CLKMT0_RMN divided by the factor shown in CMU_CSR[RCDIV], and reports this event. The CMU can also monitor CLKMN1 and generate an event if CLKMN1 is greater than a high frequency boundary or less than a low frequency boundary. The upper and lower frequency boundaries are defined by the CMU High Frequency Reference Register (CMU_HFREFR) and CMU Low Frequency Reference Register (CMU_LFREFR).

The second task of the CMU is to provide a frequency meter, which allows measuring the frequency of a clock source against a reference clock. This is useful to allow the calibration of the metered clocks (such as CLKMT0_RMN, CLKMT1, CLKMT2), as well as to be able to correct/calculate the time deviation of a counter that is clocked by the metered clocks.

NOTE

See the "Clocking" chapter for the number of CMU instances on this chip.

30.1.1 Main features

- CLKMT0_RMN, CLKMT1, CLKMT2 frequency measurement with CLKMN0_RMT as reference clock.
- CLKMN0_RMT monitoring with respect to CLKMT0_RMN ÷ $2^{CSR[RCDIV]}$ clock.
- Upper or lower frequency boundary monitoring of CLKMN1 with respect to CLKMT0_RMN ÷ 4.
- Event generation for various failures detected inside monitoring unit.

30.2 Block diagram

The block diagram of the CMU module(s) is shown in the “Clocking" chapter of this Reference Manual.

30.3 Signals

The table below describes the signals on the boundary of the CMU (in alphabetical order).

Table 30-1. Signal description

Signal	I/O	Description
CLKMN0_RMT	I	Monitored Clock Signal 0/Metered Clock Signal Reference — Receives a clock signal that the CMU compares to a specified low-limit frequency to determine whether the frequency of the clock signal is greater than the specified limit. Also provides a reference clock signal for all metered clock signals.
CLKMN1	I	Monitored Clock Signal 1 — Receives a clock signal that the CMU compares to specified low-limit and high-limit frequencies to determine whether the frequency of the clock signal is between the specified limits.

Table continues on the next page...

Table 30-1. Signal description (continued)

Signal	I/O	Description
CLKMT0_RMN	I	Metered Clock Signal 0/Monitored Clock Signal Reference — Receives a clock signal that the CMU measures against a reference clock frequency. Also provides a reference clock signal for all monitored clock signals.
CLKMT1	I	Metered Clock Signal 1 — Receives a clock signal that the CMU measures against a reference clock frequency.
CLKMT2	I	Metered Clock Signal 2 — Receives a clock signal that the CMU measures against a reference clock frequency.

NOTE

See the "Clocking" chapter for device specific clock sources of each CMU.

30.4 Register description and memory map

This section describes in address order all the CMU registers. Each description includes a standard register diagram with an associated figure number. The CMU memory map is listed in the following table.

NOTE

See "Clocking" chapter for register and field availability details.

CMU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	CMU Control Status Register (CMU_CSR)	32	R/W	See section	30.4.1/624
4	CMU Frequency Display Register (CMU_FDR)	32	R	0000_0000h	30.4.2/625
8	CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)	32	R/W	0000_0FFFh	30.4.3/626
C	CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)	32	R/W	0000_0000h	30.4.4/626
10	CMU Interrupt Status Register (CMU_ISR)	32	w1c	See section	30.4.5/627
18	CMU Measurement Duration Register (CMU_MDR)	32	R/W	0000_0000h	30.4.6/629

30.4.1 CMU Control Status Register (CMU_CSR)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0								SFM	0						
W																
Reset	0	0	0	0	0	0	0	0	0*	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						CKSEL1		0					RCDIV		CME
W																
Reset	0	0	0	0	0	0	0*	0*	0	0	0	0	0	0*	0*	0

* Notes:
- RCDIV field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- CKSEL1 field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.
- SFM field: Not all CMU blocks will utilize this feature. See the “Clocking” chapter for CMU implementation details.

CMU_CSR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 SFM	Start frequency measure. The software can only set this bit to start a clock frequency measure. It is reset by hardware when the measure is ready in the CMU_FDR. **NOTE:** MSR[MD] must be written before enabling frequency measurement (CSR[SFM] = 1). Do not write MDR[MD] while CSR[SFM] = 1. 0 Frequency measurement is completed or not yet started 1 Frequency measurement is not completed
9–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 CKSEL1	Frequency measure clock selection bit. CKSEL1 selects the clock to be measured by the frequency meter. This only affects CMU instances that utilizes clock metering. Not all CMU blocks will utilize this feature. See the “Clocking” chapter for device specific CMU implementation details. 00 CLKMT0_RMN is selected 01 CLKMT1 is selected 10 CLKMT2 is selected 11 CLKMT0_RMN is selected
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

CMU_CSR field descriptions (continued)

Field	Description
29–30 RCDIV	CLKMT0_RMN division factor. These bits specify the CLKMT0_RMN division factor. The output clock frequency is $f_{CLKMT0_RMN} \div 2^{CMU_CSR[RCDIV]}$. This output clock is used as reference clock to compare with CLKMN0_RMT for crystal clock monitor feature. 00 CLKMT0_RMN ÷ 1 (No division) 01 CLKMT0_RMN ÷ 2 10 CLKMT0_RMN ÷ 4 11 CLKMT0_RMN ÷ 8
31 CME	CLKMN1 monitor enable. 0 CLKMN1 monitor is disabled 1 CLKMN1 monitor is enabled

30.4.2 CMU Frequency Display Register (CMU_FDR)

The CMU_FDR is used to determine the measured frequency of:

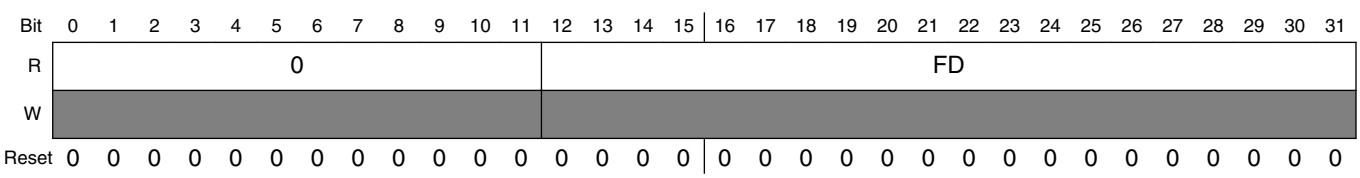
- CLKMT0_RMN
- CLKMT2
- CLKMT1

with respect to the reference clock CLKMN0_RMT.

NOTE

The CMU_FDR should be read only when CMU_CSR[SFM] = 0.

Address: 0h base + 4h offset = 4h



CMU_FDR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 FD	Measured frequency bits. This register displays the measured frequency (f_{sel}) with respect to the reference clock (f_{CLKMN0_RMT}). The measured value is given by the following formula: $f_{sel} = (f_{CLKMN0_RMT} \times CMU_MDR[MD]) \div CMU_FDR[FD]$

30.4.3 CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)

The HFREFR is configured for the high frequency reference that the CMU will use for comparing against the monitored clock. The figure and table below show the CMU_HFREFR register.

NOTE

For correct synchronization of CMU_HFREFR write data, the user must ensure that a time interval of 10 peripheral bus clock cycles + 40 CLKMT0_RMN clock cycles elapses before attempting subsequent writes to the same register.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																0				HFREF											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

CMU_HFREFR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 HFREF	High Frequency reference value. These bits determine the high reference value for the CLKMN1 frequency. The reference value is given by: (HFREF ÷ 16) × (f_{CLKMT0_RMN} ÷ 4).

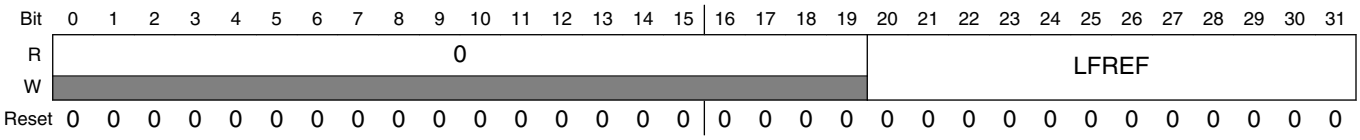
30.4.4 CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

The LFREFR is configured for the low frequency reference that the CMU will use for comparing against the monitored clock. The figure and table below show the CMU_LFREFR register.

NOTE

For correct synchronization of CMU_LFREFR write data, the user must ensure that a time interval of 10 peripheral bus clock cycles + 40 CLKMT0_RMN clock cycles elapses before attempting subsequent writes to the same register.

Address: 0h base + Ch offset = Ch



CMU_LFREFR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 LFREF	Low Frequency reference value. These bits determine the low reference value for the CLKMN1 frequency. The reference value is given by: (LFREF ÷ 16) × (f_{CLKMT0_RMN} ÷ 4).

30.4.5 CMU Interrupt Status Register (CMU_ISR)

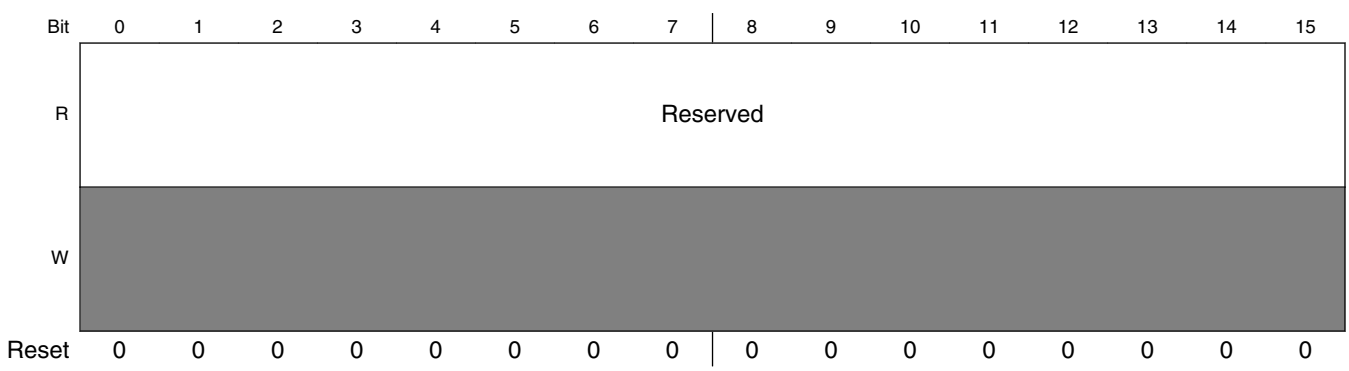
NOTE

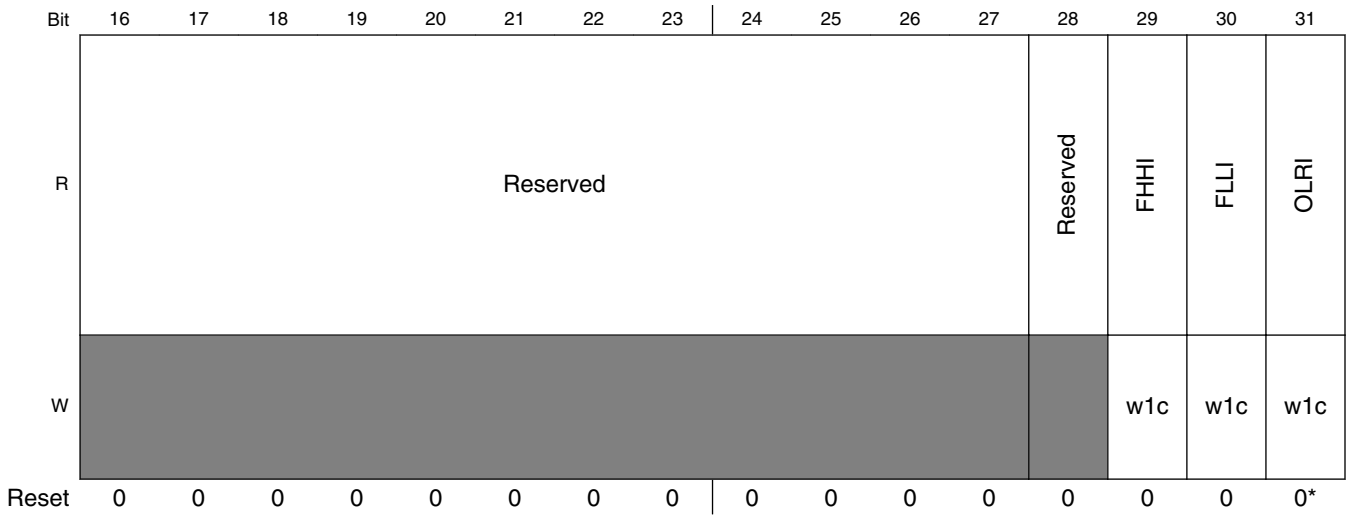
All flags in the CMU_ISR are set asynchronously. This register must be read only after a CMU event is triggered. Otherwise, a read access on this register may fetch an incorrect value (see "Clocking" chapter for interrupt operation) .

NOTE

Before entering low-power stop modes, all clock frequency measurements in the CMU should be disabled. Failing to do so may result in ISR flags being spuriously set.

Address: 0h base + 10h offset = 10h





* Notes:
- OLRI field: Not all CMU blocks will utilize this feature. See the "Clocking" chapter for specific CMU implementation details.

CMU_ISR field descriptions

Field	Description
0–27 Reserved	This field is reserved.
28 Reserved	This field is reserved. **NOTE:** This bit will show indeterministic behavior if the clock monitored by the CMU is not enabled .
29 FHHI	CLKMN1 frequency higher than high reference event status. This bit is set by hardware when CLKMN1 frequency becomes higher than the high frequency reference value determined by CMU_HFREFR[HFREF] and CLKMN1 is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. 0 No FHH event 1 FHH event occurred
30 FLLI	CLKMN1 frequency less than low reference event status. This bit is set by hardware when CLKMN1 frequency becomes lower than the low frequency reference value determined by CMU_LFREFR[LFREF], and CLKMN1 is 'ON' as signaled by the MC_ME. Software clears this field by writing a '1'. **NOTE:** This bit will show indeterministic behavior if the clock monitored by the CMU is not enabled . 0 No FLL event 1 FLL event occurred
31 OLRI	Oscillator frequency less than $f_{CLKMT0_RMN} \div 2^{CMU_CSR[RCDIV]}$event status. This bit is set by hardware when the f_{CLKMN0_RMT} is less than $f_{CLKMT0_RMN} \div 2^{CMU_CSR[RCDIV]}$frequency and CLKMN0_RMT is 'ON' as signaled by the MC_ME. It can be cleared by software by writing '1'. **NOTE:** While entering STOP mode, OLRI may be triggered due to stoppage of XOSC. To avoid this, disable the XOSC before entering STOP mode.

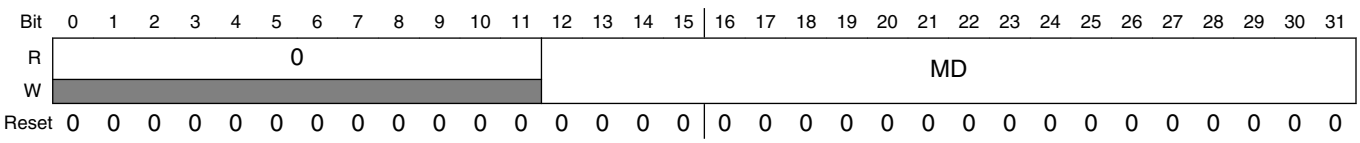
Table continues on the next page...

CMU_ISR field descriptions (continued)

Field	Description
	0 No OLR event 1 OLR event occurred

30.4.6 CMU Measurement Duration Register (CMU_MDR)

Address: 0h base + 18h offset = 18h



CMU_MDR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 MD	Measurement duration bits This field displays the measurement duration in terms of selected clock (CLKMT0_RMN, CLKMT1, CLKMT2) cycles. This value is loaded in the frequency meter down-counter. The down-counter starts counting when CMU_CSR [SFM] = 1. **NOTE:** Only values that do not cause a CMU_FDR[FD] overflow shall be written to CMU_MDR[MD]. There is no internal status of a CMU_FDR[FD] overflow.

30.5 Functional description

30.5.1 Frequency meter

The purpose of the frequency meter is to evaluate the deviation from the nominal metered source (such as CLKMT0_RMN, CLKMT2, CLKMT1) frequencies. This in turn allows either recalibration of these clocks or other timing corrections. Programming of the CMU_CSR[CKSEL1] field is used to select one of the metered clocks from a multiplexer that drives a simple Frequency Meter (see the CMU Block Diagram in the "Clocking" chapter). The reference clock for the Frequency Meter is the CLKMN0_RMT signal. The measurement starts when CMU_CSR[SFM] = 1. The measurement duration is given by the contents of CMU_MDR[MD] in terms of number of clock cycles of the selected metered clock. The CMU_CSR[SFM] bit is cleared by hardware once the frequency

measurement is complete and the count is loaded in CMU_FDR[FD]. The frequency of the selected clock (f_{sel}) can be derived from the value loaded in the CMU_FDR[FD] as shown in the following equation:

$$f_{sel} = f_{CLKMN0_RMT} \times \frac{CMU_MDR[MD]}{CMU_FDR[FD]}$$

30.5.2 CLKMN0_RMT supervisor

If frequency of CLKMN0_RMT is smaller than the frequency of CLKMT0_RMN $\div 2^{CMU_CSR[RCDIV]}$ and CLKMN0_RMT is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[OLRI].
- The CMU asserts the OLR signal.

Note

f_{CLKMN0_RMT} must be greater than $f_{CLKMT0_RMN} \div 2^{CMU_CSR[RCDIV]}$ by at least 0.5 MHz in order to guarantee correct f_{CLKMN0_RMT} monitoring.

30.5.3 CLKMN1 supervisor

The frequency of CLKMN1(f_{CLKMN1}) can be monitored by programming CMU_CSR[CME] = 1. CLKMN1 monitoring starts as soon as CMU_CSR[CME] = 1. This monitor can be disabled at any time by programming CMU_CSR[CME] = 0.

If f_{CLKMN1} is greater than the reference value determined by fields CMU_HFREFR[HFREF] and CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FHHI].
- The CMU asserts the FHH signal.

If f_{CLKMN1} is less than a reference value determined by the bits CMU_LFREFR[LFREF] and the CLKMN1 is 'ON' as signaled by the MC_ME, then:

- The CMU writes 1 to CMU_ISR[FLLI].
- The CMU asserts the FLL signal.

Note

An example of determining the $HFREF_{Actual}$ is as follows. Assume a f_{CLKMT0_RMN} = 16 MHz with a accuracy of +/-5% (see *Data Sheet* for actual value). In order to monitor a f_{CLKMN1} = 200 MHz, the ideal $HFREF_{Ideal}$ = 800. The actual HFREF value will be 842 when the accuracy is taken into consideration ($HFREF_{Actual}$ = ($HFREF_{Ideal}$ ÷ 0.95). The actual LFREF value will be 762 when accuracy is taken into consideration ($LFREF_{Actual}$ = $LFREF_{Ideal}$ ÷ 1.05).

Chapter 31
Clock Generation Module (MC_CGM)

31.1 Introduction

31.1.1 Overview

The clock generation module (MC_CGM) generates reference clocks for all the chip blocks. The MC_CGM selects one of the system clock sources to supply the system clock. The MC_ME controls the system clock selection (see the MC_ME chapter for more details). The memory spaces of system and peripheral clock sources which have addressable memory spaces are accessed through the MC_CGM memory space.

The following figure shows the MC_CGM block diagram.

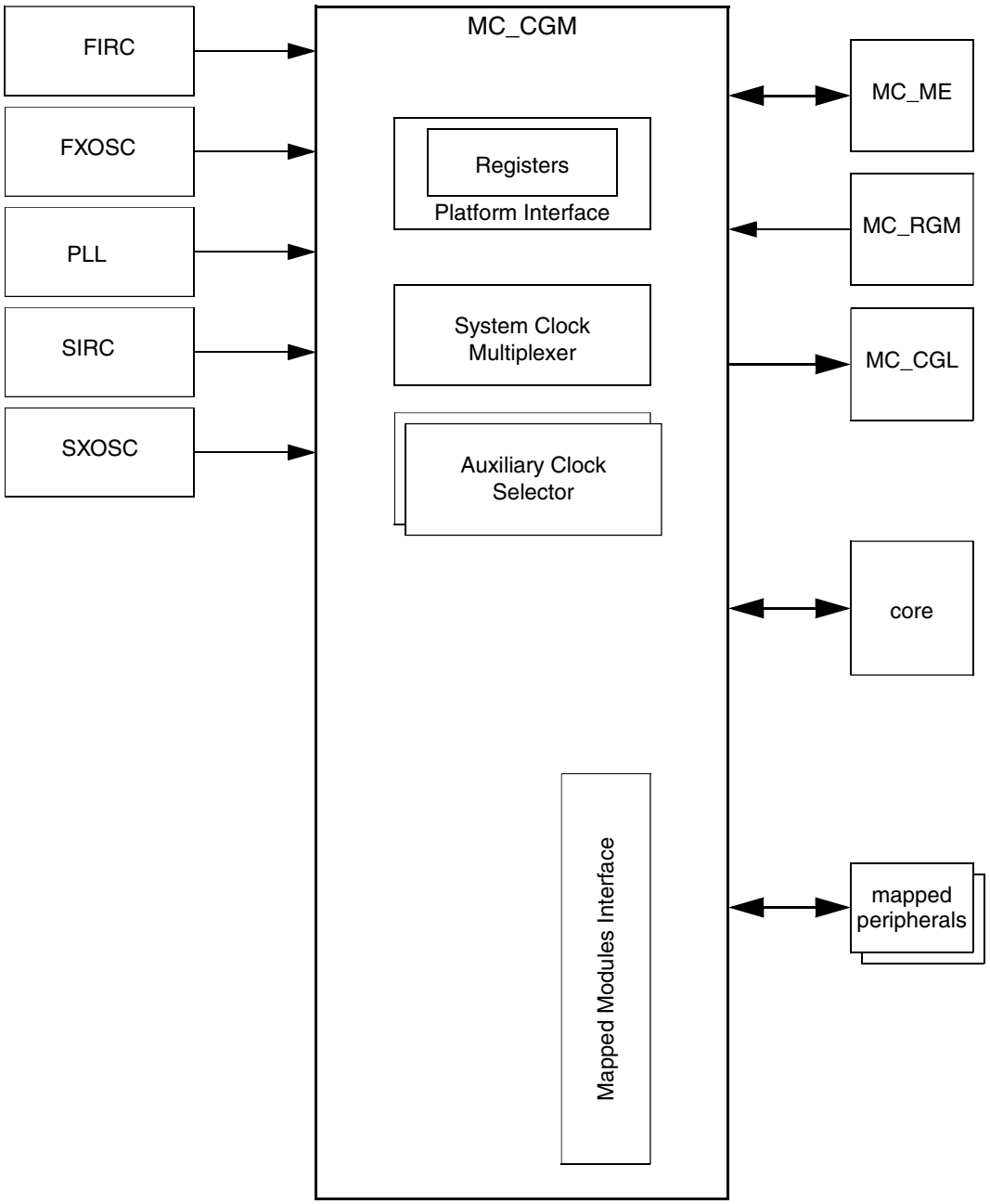


Figure 31-1. MC_CGM Block Diagram

31.1.2 Features

The MC_CGM includes the following features:

- generates system and peripheral clocks
- selects and enables/disables the system clock supply from system clock sources according to MC_ME control

- performs progressive system clock frequency change depending on MC_ME mode configuration
- contains a set of registers to control clock dividers for divided clock generation
- contains a set of registers to control peripheral clock selection
- contains IPS interface for mapped clock source modules
- supports multiple clock sources and maps their address spaces to its memory map
- supports 8, 16, and 32-bit wide read/write accesses

31.2 External Signal Description

The MC_CGM delivers output clocks to the CLKOUT0 and CLKOUT1 pins for off-chip use and/or observation.

31.3 Memory Map and Register Definition

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the CGM_PCS_DIVC1 register RATE bits may be accessed as a word at address offset 0x0704, as a half-word at address offset 0x0706, or as a byte at address offset 0x0707.

MC_CGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
180	Clockout 1 Select Register (MC_CGM_CLKOUT1_SC)	32	R/W	0000_0000h	31.3.1/637
188	CLKOUT1_DC Configuration Register (MC_CGM_CLKOUT1_DC)	32	R/W	8000_0000h	31.3.2/638
1C0	Auxiliary Clock 5 Select Control Register (MC_CGM_AC5_SC)	32	R/W	0000_0000h	31.3.3/639
1C4	Auxiliary Clock 5 Select Status Register (MC_CGM_AC5_SS)	32	R/W	0000_0000h	31.3.4/640
200	System Clock Divider 6 Configuration Register (MC_CGM_SC_DC6)	32	R	8007_0000h	31.3.5/641
280	Auxiliary Clock 8 Select Control Register (MC_CGM_AC8_SC)	32	R/W	0000_0000h	31.3.6/641

Table continues on the next page...

MC_CGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
284	Auxiliary Clock 8 Select Status Register (MC_CGM_AC8_SS)	32	R/W	0000_0000h	31.3.7/642
2C0	Auxiliary Clock 9 Select Control Register (MC_CGM_AC9_SC)	32	R/W	0000_0000h	31.3.8/643
2C4	Auxiliary Clock 9 Select Status Register (MC_CGM_AC9_SS)	32	R/W	0000_0000h	31.3.9/644
700	PCS Switch Duration Register (MC_CGM_PCS_SDUR)	8	R/W	00h	31.3.10/ 645
704	PCS Divider Change Register 1 (MC_CGM_PCS_DIVC1)	32	R/W	03E7_0000h	31.3.11/ 646
708	PCS Divider End Register 1 (MC_CGM_PCS_DIVE1)	32	R/W	0000_03E7h	31.3.12/ 646
70C	PCS Divider Start Register 1 (MC_CGM_PCS_DIVS1)	32	R/W	0000_03E7h	31.3.13/ 647
710	PCS Divider Change Register 2 (MC_CGM_PCS_DIVC2)	32	R/W	03E7_0000h	31.3.14/ 648
714	PCS Divider End Register 2 (MC_CGM_PCS_DIVE2)	32	R/W	0000_03E7h	31.3.15/ 648
718	PCS Divider Start Register 2 (MC_CGM_PCS_DIVS2)	32	R/W	0000_03E7h	31.3.16/ 649
7CC	Divider Update Abort Register (MC_CGM_DIV_UPD_ABRTD)	32	R/W	0000_0001h	31.3.17/ 650
7D0	System Clock Divider Ratio Change Register (MC_CGM_SC_DIV_RC)	32	R/W	0000_0001h	31.3.18/ 651
7D4	Divider Update Type Register (MC_CGM_DIV_UPD_TYPE)	32	R/W	0000_0000h	31.3.19/ 653
7D8	Divider Update Trigger Register (MC_CGM_DIV_UPD_TRIG)	32	R/W	0000_0000h	31.3.20/ 654
7DC	Divider Update Status Register (MC_CGM_DIV_UPD_STAT)	32	R/W	0000_0000h	31.3.21/ 655
7E4	System Clock Select Status Register (MC_CGM_SC_SS)	32	R	0008_0000h	31.3.22/ 656
7E8	System Clock Divider 0 Configuration Register (MC_CGM_SC_DC0)	32	R/W	8000_0000h	31.3.23/ 657
7EC	System Clock Divider 1 Configuration Register (MC_CGM_SC_DC1)	32	R/W	8001_0000h	31.3.24/ 658
7F0	System Clock Divider 2 Configuration Register (MC_CGM_SC_DC2)	32	R/W	8003_0000h	31.3.25/ 659
7F4	System Clock Divider 3 Configuration Register (MC_CGM_SC_DC3)	32	R	8003_0000h	31.3.26/ 660
7F8	System Clock Divider 4 Configuration Register (MC_CGM_SC_DC4)	32	R	8001_0000h	31.3.27/ 661
7FC	System Clock Divider 5 Configuration Register (MC_CGM_SC_DC5)	32	R/W	8001_0000h	31.3.28/ 662

Table continues on the next page...

MC_CGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
840	Auxiliary Clock 2 Select Control Register (MC_CGM_AC2_SC)	32	R/W	0000_0000h	31.3.29/ 663
844	Auxiliary Clock 2 Select Status Register (MC_CGM_AC2_SS)	32	R/W	0000_0000h	31.3.30/ 664
864	Auxiliary Clock 3 Select Status Register (MC_CGM_AC3_SS)	32	R	0000_0000h	31.3.31/ 665
8C0	Auxiliary Clock 6 Select Control Register (MC_CGM_AC6_SC)	32	R/W	0000_0000h	31.3.32/ 665
8C4	Auxiliary Clock 6 Select Status Register (MC_CGM_AC6_SS)	32	R	0000_0000h	31.3.33/ 666
8C8	Aux Clock 6 Divider Configuration 0 for CLKOUT_0 (MC_CGM_AC6_DC0)	32	R/W	8000_0000h	31.3.34/ 667

31.3.1 Clockout 1 Select Register (MC_CGM_CLKOUT1_SC)

The CLK_CLKOUT1_SC is used to choose the clock source for the CLKOUT1 output.

NOTE

CLKOUT1 muxing is available in always alive domain and clock can be observed in all LPU modes and STANDBY mode. Refer General Programmable Registers (GPR) chapter for more configurations.

This register is reset only on POR reset.

Address: 0h base + 180h offset = 180h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_CLKOUT1_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	CLKOUT1 Source Select SELCTL selects the source for the CLKOUT1 output. 0000 FXOSC divided 0001 FIRC divided 0010 SXOSC 0011 SIRC

Table continues on the next page...

MC_CGM_CLKOUT1_SC field descriptions (continued)

Field	Description
	0100 RTC clk 0101 LPU sys clk 0110 Z2 core clk 0111 FXOSC ANA Clk 1000 FXOSC 1001 FIRC 1010 SXOSC divided 1011 SIRC divided 1100 CAN0 CHI clk 1101 CAN0 PE clk 1110 PLL_CLOCKOUT1 1111 Reserved
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.2 CLKOUT1_DC Configuration Register (MC_CGM_CLKOUT1_DC)

This register corresponds to CLKOUT1_DC system clock divider.

This register is reset only on POR reset, while on PHASE3 reset the clock divider loads the default reset value at the completion of division cycle corresponding to programmed value.

Address: 0h base + 188h offset = 188h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0				DIV										
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_CLKOUT1_DC field descriptions

Field	Description
0 DE	CLKOUT1_DC enable 0 Divider disabled 1 Divider enabled
1–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_CGM_CLKOUT1_DC field descriptions (continued)

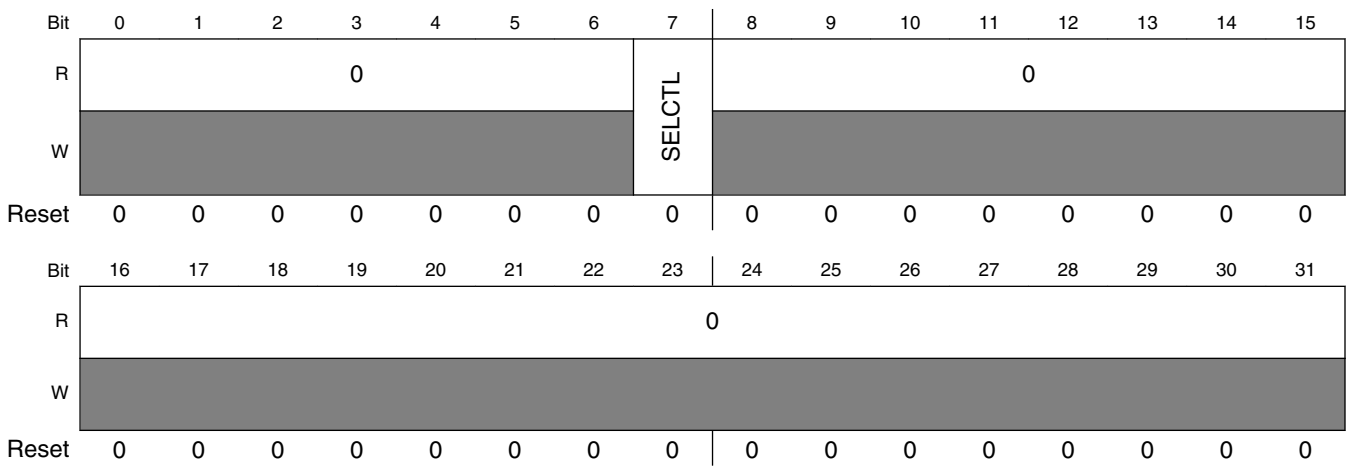
Field	Description
5–15 DIV	Divider CLKOUT1_DC Division Value - The resultant CLKOUT1_DC clock will have a period 'DIV + 1' times that of the source clock. If DE is set to '0' (divider is disabled), any write access to the DIV field is ignored and the clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.3 Auxiliary Clock 5 Select Control Register (MC_CGM_AC5_SC)

This register is used to select the current clock source for the PLL Aux clock.

This register is reset only on POR reset.

Address: 0h base + 1C0h offset = 1C0h



MC_CGM_AC5_SC field descriptions

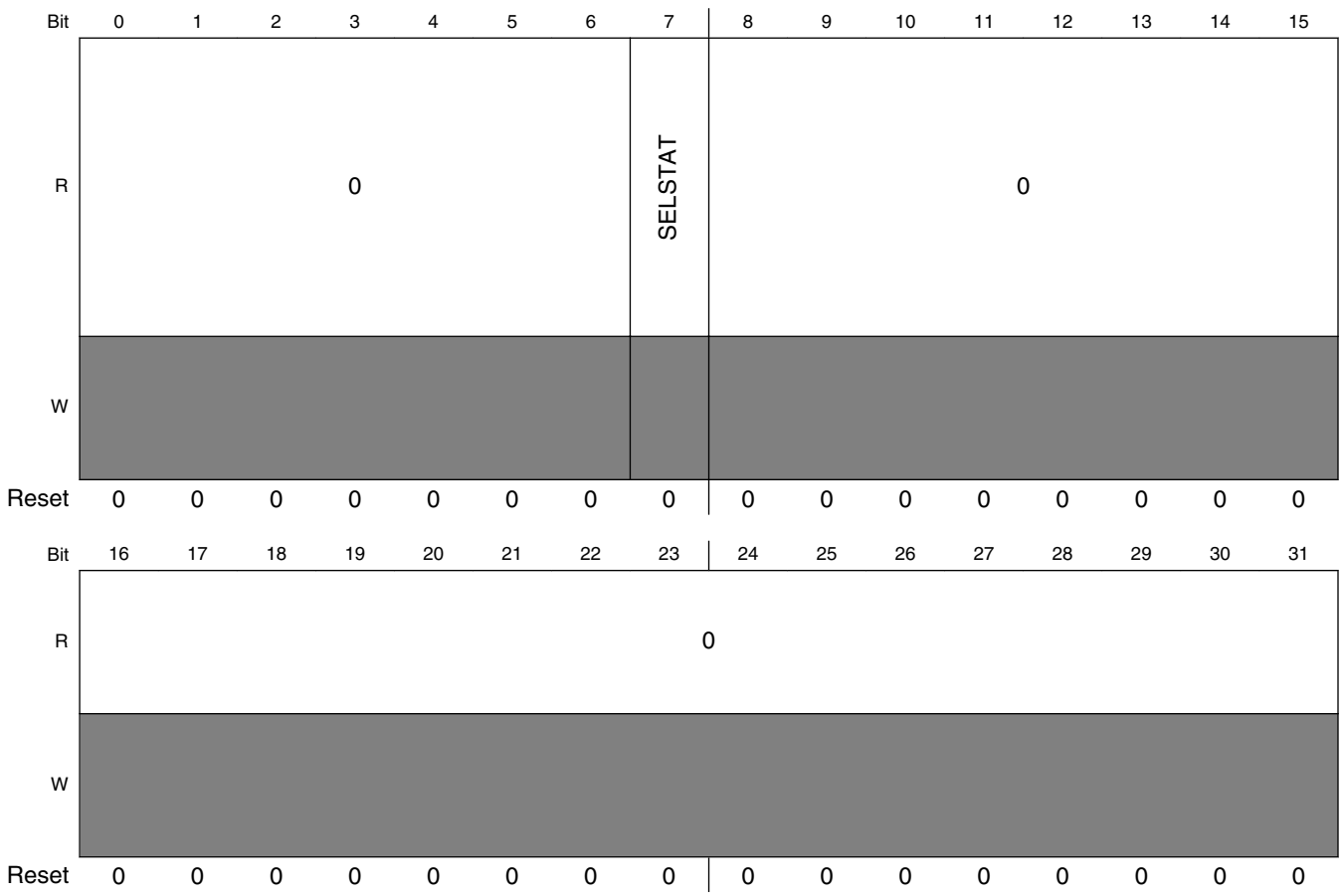
Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELCTL	Auxiliary Clock 5 Source Selection Control — Selects the source for auxiliary clock 5. 0 Fast Internal crystal osc. (FIRC) 1 Fast external crystal osc. (FXOSC)
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.4 Auxiliary Clock 5 Select Status Register (MC_CGM_AC5_SS)

This register provides the current auxiliary clock 5 source selection

This register is reset only on POR reset.

Address: 0h base + 1C4h offset = 1C4h



MC_CGM_AC5_SS field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELSTAT	Auxiliary Clock 5 Source Selection Status — This value indicates the current source for auxiliary clock 5. 0 Fast Internal crystal osc. (FIRC) 1 Fast external crystal osc. (FXOSC)
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.5 System Clock Divider 6 Configuration Register (MC_CGM_SC_DC6)

This register corresponds to F20 system clock divider.

NOTE

This divider should never be configured to change the default value. Refer Clocking chapter for details.

This register is reset only on POR reset, while on PHASE3 reset the clock divider loads the default reset value at the completion of division cycle corresponding to programmed value.

Address: 0h base + 200h offset = 200h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0												DIV		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_SC_DC6 field descriptions

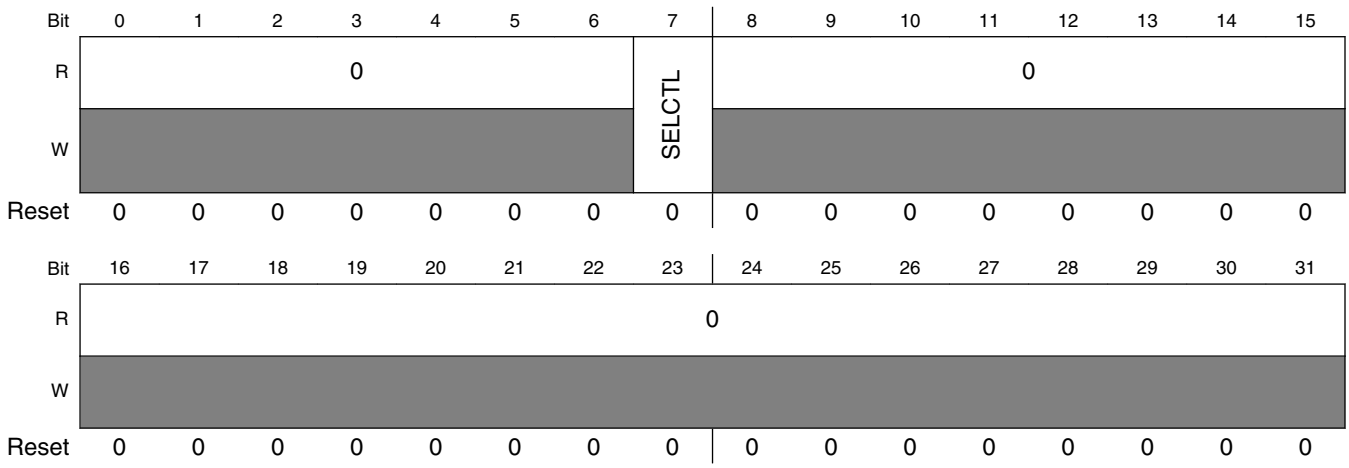
Field	Description
0 DE	Divider 6 enable 0 Divider disabled
1–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 DIV	Divider Divide 6 Division Value - The resultant F20 clock will have a period ‘DIV + 1’ times that of the source clock. If DE is set to ‘0’ (divider is disabled), any write access to the DIV field is ignored and the clock remains disabled.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.6 Auxiliary Clock 8 Select Control Register (MC_CGM_AC8_SC)

This register is used to select the current clock source for the SPI0.

This register is reset only on POR reset.

Address: 0h base + 280h offset = 280h



MC_CGM_AC8_SC field descriptions

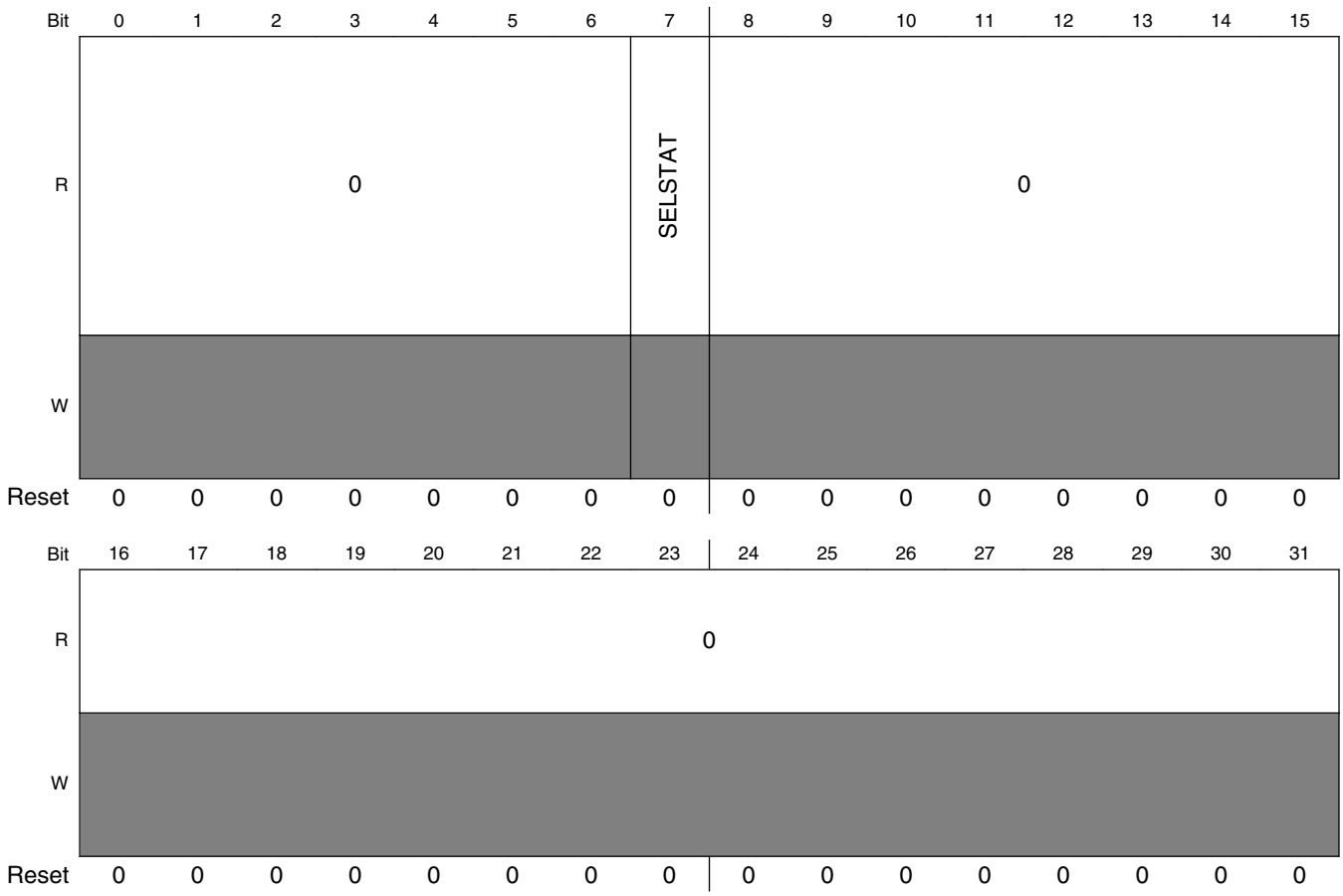
Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELCTL	Auxiliary Clock 8 Source Selection Control — This value selects the current source for auxiliary clock 8. 0 F40 1 FXOSC
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.7 Auxiliary Clock 8 Select Status Register (MC_CGM_AC8_SS)

This register provides the current auxiliary clock 8 source selection for SPI0.

This register is reset only on POR reset.

Address: 0h base + 284h offset = 284h



MC_CGM_AC8_SS field descriptions

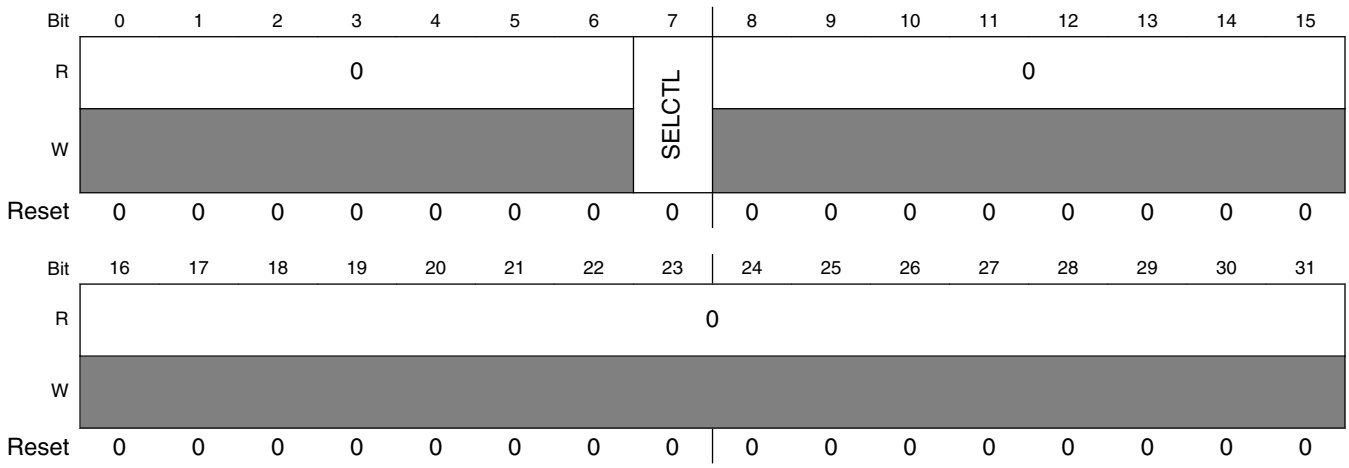
Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELSTAT	Auxiliary Clock 8 Source Selection Status — This value indicates the current source for auxiliary clock 8. 0 F40 1 FXOSC
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.8 Auxiliary Clock 9 Select Control Register (MC_CGM_AC9_SC)

This register is used to select the current clock source for the FlexCAN0.

This register is reset only on POR reset.

Address: 0h base + 2C0h offset = 2C0h



MC_CGM_AC9_SC field descriptions

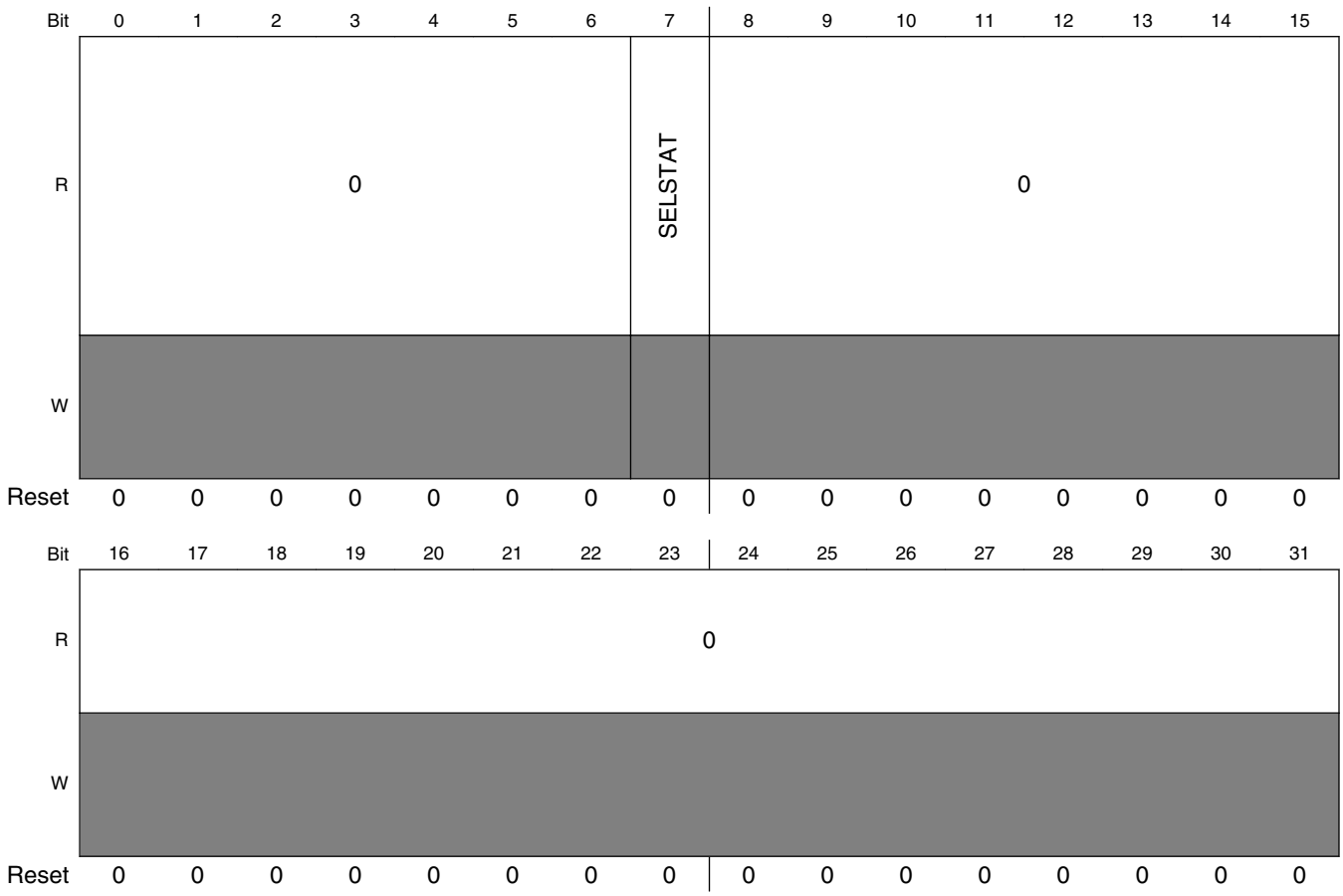
Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELCTL	Auxiliary Clock 9 Source Selection Control — This value selects the current source for auxiliary clock 9. 0 FS80 1 FXOSC
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.9 Auxiliary Clock 9 Select Status Register (MC_CGM_AC9_SS)

This register provides the current auxiliary clock 9 source selection for FlexCAN0.

This register is reset only on POR reset.

Address: 0h base + 2C4h offset = 2C4h



MC_CGM_AC9_SS field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELSTAT	Auxiliary Clock 9 Source Selection Status — This value indicates the current source for auxiliary clock 9. 0 FS80 1 FXOSC
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.10 PCS Switch Duration Register (MC_CGM_PCS_SDUR)

This register contains the progressive system clock switching duration for each step. See Progressive System Clock Switching for details on how to set this value.

This register is reset only on POR reset.

Address: 0h base + 700h offset = 700h

Bit	0	1	2	3	4	5	6	7
Read Write	SDUR							
Reset	0	0	0	0	0	0	0	0

MC_CGM_PCS_SDUR field descriptions

Field	Description
0–7 SDUR	Switch Duration This value defines the duration of one PCS clock switch step in terms of 16 MHz int. RC osc. cycles.

31.3.11 PCS Divider Change Register 1 (MC_CGM_PCS_DIVC1)

This register defines the rate of frequency change and initial change value for the progressive system clock switching when switching the system clock source to or from the fast external crystal oscillator. See Progressive System Clock Switching for details on how to set these values.

This register is reset only on POR reset.

Address: 0h base + 704h offset = 704h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INIT																0								RATE							
W																																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_PCS_DIVC1 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value This is initial change value of the clock divider for the clock ramp-up phase when switching to fast external crystal oscillator
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching from the fast external crystal oscillator

31.3.12 PCS Divider End Register 1 (MC_CGM_PCS_DIVE1)

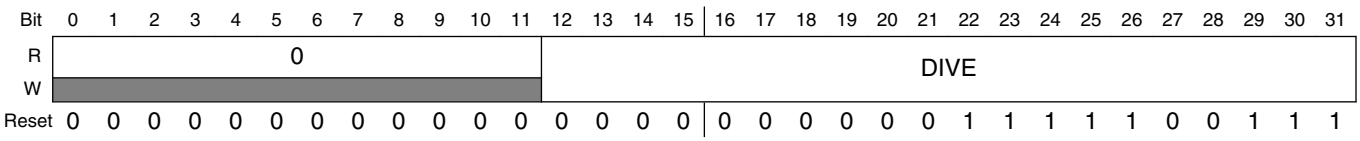
This register defines the final division value for the progressive system clock switching when switching from or to fast external crystal oscillator on ramp up and ramp down. See Progressive System Clock Switching for details on how to set this value.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

This register is reset only onPOR reset.

Address: 0h base + 708h offset = 708h



MC_CGM_PCS_DIVE1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value - This is the clock divider end value for the clock ramp-down phase when switching from the fast external crystal oscillator.

31.3.13 PCS Divider Start Register 1 (MC_CGM_PCS_DIVS1)

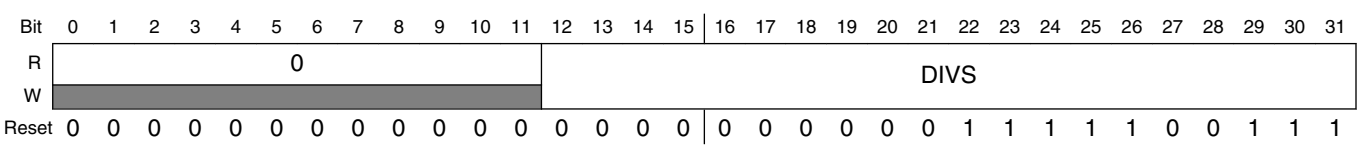
These registers define the initial division value on frequency ramp-up for the progressive system clock switching when switching the system clock from or to the fast external crystal oscillator on ramp up and ramp down. *n* . See Progressive System Clock Switching for details on how to set these values.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

This register is reset only on POR reset.

Address: 0h base + 70Ch offset = 70Ch



MC_CGM_PCS_DIVS1 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_CGM_PCS_DIVS1 field descriptions (continued)

Field	Description
12–31 DIVS	Divider Start Value This is the start value of the clock divider for the clock ramp-up phase when switching to the fast external crystal oscillator

31.3.14 PCS Divider Change Register 2 (MC_CGM_PCS_DIVC2)

These registers define the rate of frequency change and initial change value on frequency ramp-up for the progressive system clock switching when switching to or from the PLL PHI0. See Progressive System Clock Switching for details on how to set these values.

NOTE

Byte write accesses are not allowed for the INIT field of this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

This register is reset only on POR reset.

Address: 0h base + 710h offset = 710h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INIT																0								RATE							
W																																
Reset	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_PCS_DIVC2 field descriptions

Field	Description
0–15 INIT	Divider Change Initial Value — This is initial change value of the clock divider for the clock ramp-up phase when switching to the PLL PHI0.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RATE	Divider Change Rate — This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase when switching to/from the PLL PHI0.

31.3.15 PCS Divider End Register 2 (MC_CGM_PCS_DIVE2)

This register defines the final division value for the progressive system clock switching when switching the system clock source from the PLL PHI0 on ramp-down. See Progressive System Clock Switching for details on how to set This value.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

This register is reset only on POR reset.

Address: 0h base + 714h offset = 714h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVE																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

MC_CGM_PCS_DIVE2 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 DIVE	Divider End Value - This is the clock divider end value for the clock ramp-down phase when switching from the PLL PHI0.

31.3.16 PCS Divider Start Register 2 (MC_CGM_PCS_DIVS2)

These registers define the initial division value on frequency ramp-up for the progressive system clock switching when switching the system clock from or to the PLL PHI0 on ramp up and ramp down See Progressive System Clock Switching for details on how to set this value.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

This register is reset only on POR reset.

Address: 0h base + 718h offset = 718h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												DIVS																			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

MC_CGM_PCS_DIVS2 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_CGM_PCS_DIVS2 field descriptions (continued)

Field	Description
12–31 DIVS	Divider Start Value - This is the start value of the clock divider for the clock ramp-up phase when switching to the PLL PHI0.

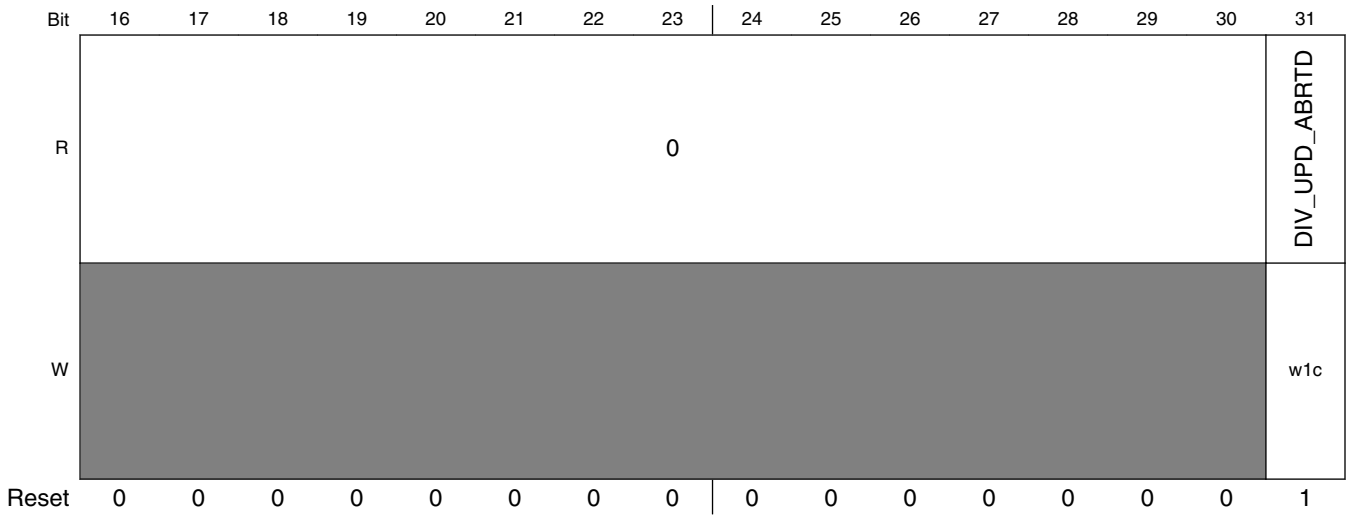
31.3.17 Divider Update Abort Register (MC_CGM_DIV_UPD_ABRTD)

This register provides the status of the halt handshake abort request, if any, from the system. Depending on when the halt abort request was received, the halt handshake may or may not be aborted. If the abort is successful and the dividers were not updated, the value of DIV_UPD_ABRTD is set as 1. In case the DIV_UPD_ABRTD bit is set, the software should clear the status by writing a 1 to bit 0 of this register and should initiate the divider update trigger at some later point. Till such a time that the dividers are updated again, the current division value of the dividers would not match the divider configuration.

This register is reset only on PHASE3 reset.

Address: 0h base + 7CCh offset = 7CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MC_CGM_DIV_UPD_ABRTD field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 DIV_UPD_ ABRTD	Divider Update Aborted **NOTE:** At power-on reset or before initiating the divider update with Halt-protocol, the DIV_UPD_ABRTD register bit should be cleared by writing 1 to it. 1 The abort is successful and the dividers were not updated, the value of DIV_UPD_ABRTD is set to 1.

31.3.18 System Clock Divider Ratio Change Register (MC_CGM_SC_DIV_RC)

This register selects whether the system clock divider ratios will change from the current configuration to the configuration in the next update. If the value of SYS_DIV_RATIO_CHNG is ‘1’ at the time of a write to the CGM_DIV_UPD_TRIG register and a pre-loaded system clock divider update is pending, the crossbar is halted while the system clock divider update takes place. If the value of SYS_DIV_RATIO_CHNG is ‘0’, the crossbar switch is not halted. This register has no effect if the system clock divider update is configured in the CGM_DIV_UPD_TYPE register to be immediate.

An example of a divider ratio not changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 4. The ratio is 1:2 in both configurations.

An example of a divider ratio changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 6. The ratio changes from 1:2 to 1:3.

The value of SYS_DIV_RATIO_CHNG is returned to ‘1’ automatically on each write to the CGM_DIV_UPD_TRIG register.

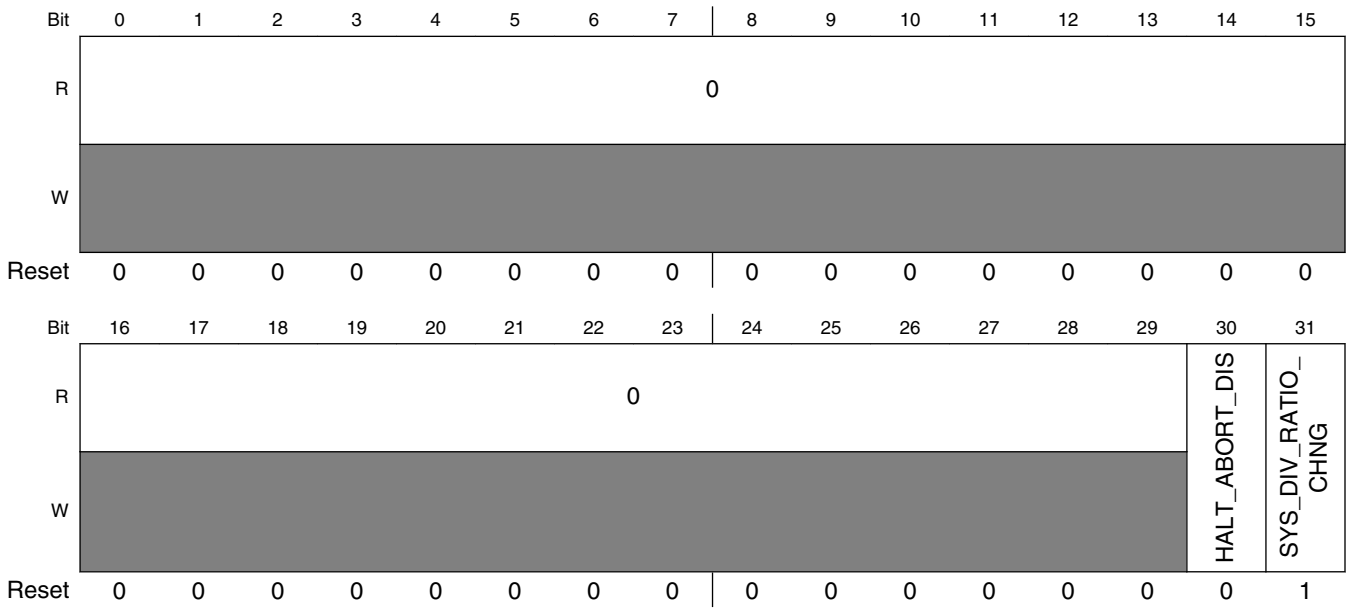
Software must never write a ‘0’ to the SYS_DIV_RATIO_CHNG bit if the system clock divider ratios change from the current configuration to the next configuration. This can potentially cause errors in the crossbar which lead to lost data.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception (i.e., ips_xfr_err is not asserted), however the value will not be loaded with the new value.

This register is reset only on PHASE3 reset.

Address: 0h base + 7D0h offset = 7D0h



MC_CGM_SC_DIV_RC field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 HALT_ABORT_DIS	Halt Abort Disable Software can disable the abort of crossbar halt handshake by setting the HALT_ABORT_DIS bit as 1. 0 Halt handshake abort is enabled 1 Halt handshake abort is disabled
31 SYS_DIV_RATIO_CHNG	System Clock Divider Ratio Change For details see, Dynamic Frequency Scaling section 0 For linear DFS, the system clock divider ratios will not change with the next system clock divider configuration update. 1 For non-linear DFS, the system clock divider ratios will change with the next system clock divider configuration update.

31.3.19 Divider Update Type Register (MC_CGM_DIV_UPD_TYPE)

This register selects whether the dividers associated with a clock are updated immediately on writing to the corresponding divider configuration register or only on writing to the divider update trigger register CGM_DIV_UPD_TRIG, after which the pre-loaded configuration in the divider configuration registers will take effect.
For usage information, see section, “Divider Update Triggering”.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

This register is reset only on PHASE3 reset.

Address: 0h base + 7D4h offset = 7D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYS_UPD_TYPE	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									Reserved	0					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_DIV_UPD_TYPE field descriptions

Field	Description
0 SYS_UPD_TYPE	System Clock Divider Update Type 0 The configuration for each system clock divider is updated immediately on writing to the corersponding CGM_SC_DCi register 1 The configuration for each system clock divider is preloaded on writing to the corersponding CGM_SC_DCi register, and the pre-loaded configurations of all the system clock dividers are updated on writing to the CGM_DIV_UPD_TRIG register
1–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 Reserved	This field is reserved.
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.20 Divider Update Trigger Register (MC_CGM_DIV_UPD_TRIG)

This register selects whether the system clock divider ratios will change from the current configuration to the configuration in the next update. If the value of SYS_DIV_RATIO_CHNG is ‘1’ at the time of a write to the CGM_DIV_UPD_TRIG register and a pre-loaded system clock divider update is pending, the crossbar is halted while the system clock divider update takes place. If the value of SYS_DIV_RATIO_CHNG is ‘0’, the crossbar switch is not halted. This register has no effect if the system clock divider update is configured in the CGM_DIV_UPD_TYPE register to be immediate.

An example of a divider ratio not changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 4. The ratio is 1:2 in both configurations.

An example of a divider ratio changing is divider 0 changes from divide by 1 to divide by 2, and divider 1 changes from divide by 2 to divide by 6. The ratio changes from 1:2 to 1:3.

The value of SYS_DIV_RATIO_CHNG is returned to ‘1’ automatically on each write to the CGM_DIV_UPD_TRIG register.

Software must never write a ‘0’ to the SYS_DIV_RATIO_CHNG bit if the system clock divider ratios change from the current configuration to the next configuration. This can potentially cause errors in the crossbar which lead to lost data.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

This register is reset only on PHASE3 reset.

Address: 0h base + 7D8h offset = 7D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	DIV_UPD_TRIGGER																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_DIV_UPD_TRIG field descriptions

Field	Description
0–31 DIV_UPD_ TRIGGER	Writing any non-zero value to this register will cause all dividers that have corresponding bits set to ‘1’ in the CGM_DIV_UPD_TYPE register and have pre-loaded configurations to be updated immediately. Reading this register will always return all zeroes.

31.3.21 Divider Update Status Register (MC_CGM_DIV_UPD_STAT)

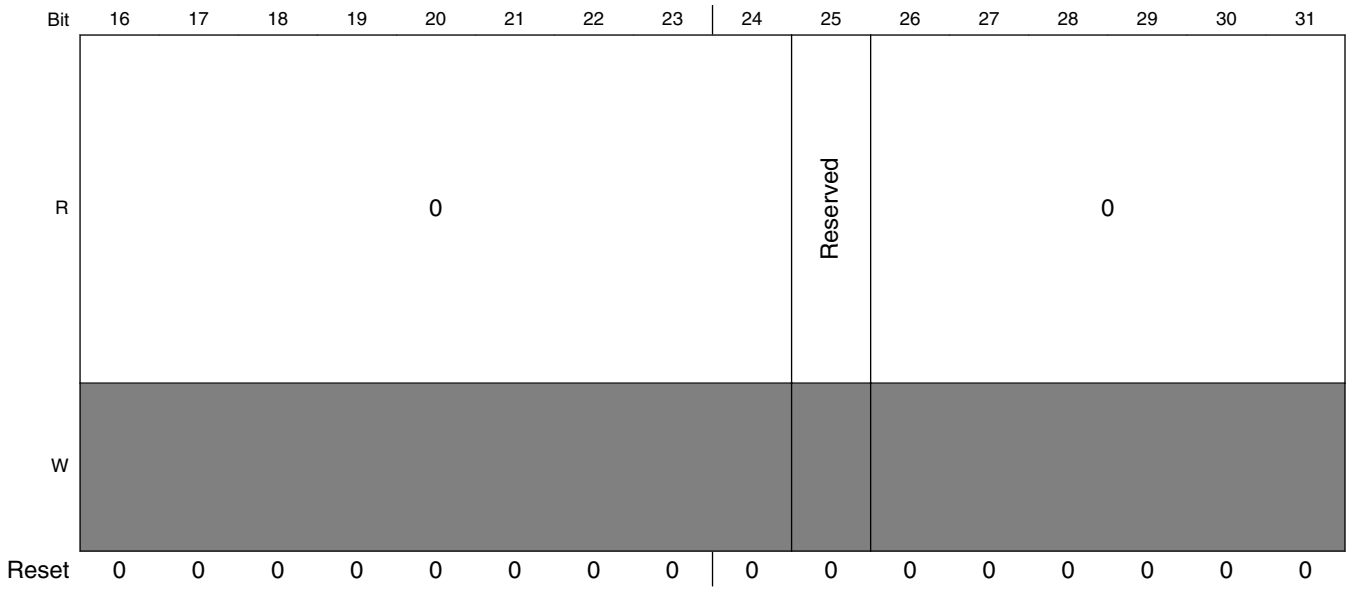
This register selects whether the dividers associated with a clock are updated immediately on writing to the corresponding divider configuration register or only on writing to the divider update trigger register CGM_DIV_UPD_TRIG, after which the pre-loaded configuration in the divider configuration registers wil take effect.

For usage information, see section, “Divider Update Triggering”.

This register is reset only on PHASE3 reset.

Address: 0h base + 7DCh offset = 7DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYS_UPD_STAT	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



MC_CGM_DIV_UPD_STAT field descriptions

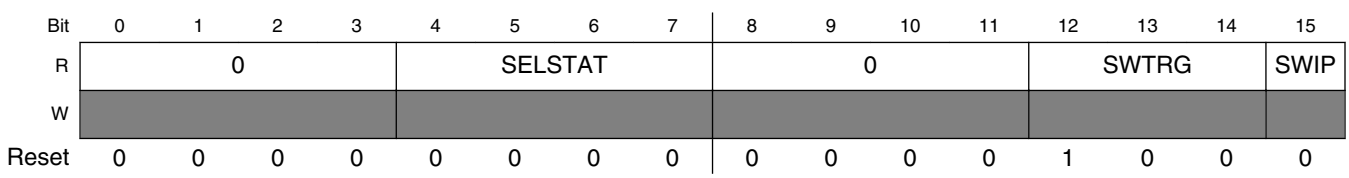
Field	Description
0 SYS_UPD_STAT	System Clock Divider Update Status 0 no system clock divider configuration updates are in process 1 the configuration for at least one system clock divider is being updated
1–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 Reserved	This field is reserved.
26–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

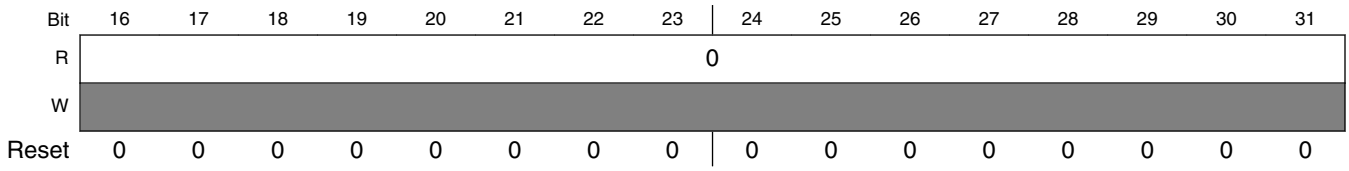
31.3.22 System Clock Select Status Register (MC_CGM_SC_SS)

This register provides the current system clock source selection.

This register is reset only on POR reset.

Address: 0h base + 7E4h offset = 7E4h





MC_CGM_SC_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	System Clock Source Selection Status - This value indicates the current source for the system clock. 0000 16 MHz internal RC osc. 0001 FXOSC 0010 PLL PHI_0 0110 reserved 0111- 1110 reserved 1111 system clock is disabled
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–14 SWTRG	Switch Trigger cause - This value indicates the cause for the latest clock source switch. 000 reserved 001 switch after request from MC_ME succeeded 010 switch after request from MC_ME failed due inactive target clock. 011 switch after request from MC_ME failed due inactive current clock [1]. 100 switch to 16 MHz int. RC osc. due to SAFE mode request or reset succeeded 101 switch to 16 MHz int. RC osc. due to SAFE mode request or reset succeeded, but current clock source was inactive 110 reserved 111 reserved
15 SWIP	Switch In Progress 0 clock source switching has completed 1 clock source switching is in progress
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

1. In such a case the system clock is switched to system clock source 0.

31.3.23 System Clock Divider 0 Configuration Register (MC_CGM_SC_DC0)

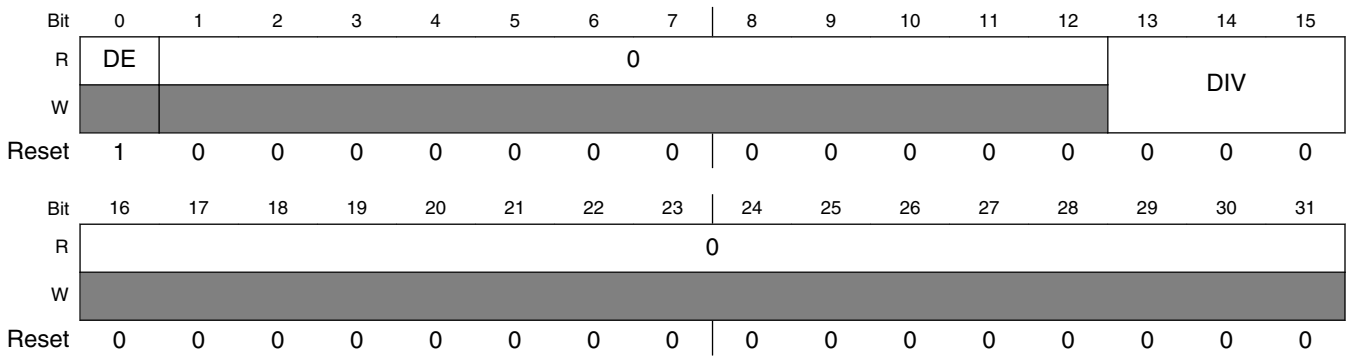
This register controls system clock divider 0 - S160 clock.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

This register is reset only on POR reset, while on PHASE3 reset the clock divider loads the default reset value at the completion of division cycle corresponding to programmed value.

Address: 0h base + 7E8h offset = 7E8h



MC_CGM_SC_DC0 field descriptions

Field	Description
0 DE	Divider 0 Enable 1 Enable system clock divider 0
1–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 DIV	Divide 0 Division Value — The resultant S160 clock will have a period ‘DIV + 1’ times that of the system clock.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.24 System Clock Divider 1 Configuration Register (MC_CGM_SC_DC1)

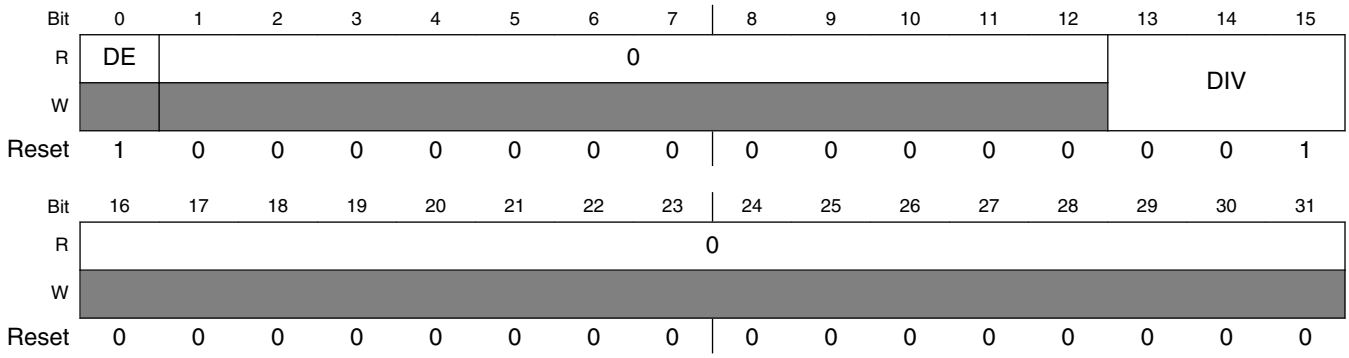
DC1 register control selected modules working on system clock divider 1 - S80 clock.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

This register is reset only on POR reset, while on PHASE3 reset the clock divider loads the default reset value at the completion of division cycle corresponding to programmed value.

Address: 0h base + 7ECh offset = 7ECh



MC_CGM_SC_DC1 field descriptions

Field	Description
0 DE	Divider 0 Enable 1 Enable system clock divider 0
1–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 DIV	Divide 1 Division Value — The resultant S80 clock will have a period ‘DIV + 1’ times that of the system clock.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.25 System Clock Divider 2 Configuration Register (MC_CGM_SC_DC2)

This register controls system clock divider 2 - S40 clock.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

This register is reset only on POR reset, while on PHASE3 reset the clock divider loads the default reset value at the completion of division cycle corresponding to programmed value.

Address: 0h base + 7F0h offset = 7F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0											DIV			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_SC_DC2 field descriptions

Field	Description
0 DE	Divider 2 Enable 1 Enable system clock divider 2
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 DIV	Divider 2 Division Value — The resultant S40 clock will have a period ‘DIV + 1’ times that of the system clock.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.26 System Clock Divider 3 Configuration Register (MC_CGM_SC_DC3)

This register controls system clock divider 3 - F40 MHz peripheral clock.

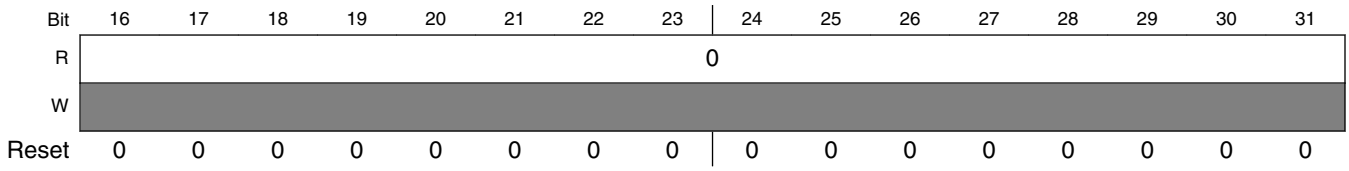
NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

This register is reset only on POR reset, while on PHASE3 reset the clock divider loads the default reset value at the completion of division cycle corresponding to programmed value.

Address: 0h base + 7F4h offset = 7F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0													DIV	
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1



MC_CGM_SC_DC3 field descriptions

Field	Description
0 DE	Divider 3 Enable 1 Enable system clock divider 3
1–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 DIV	Divider 3 Division Value — The resultant F40 clock will have a period 4 times that of system clock.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.27 System Clock Divider 4 Configuration Register (MC_CGM_SC_DC4)

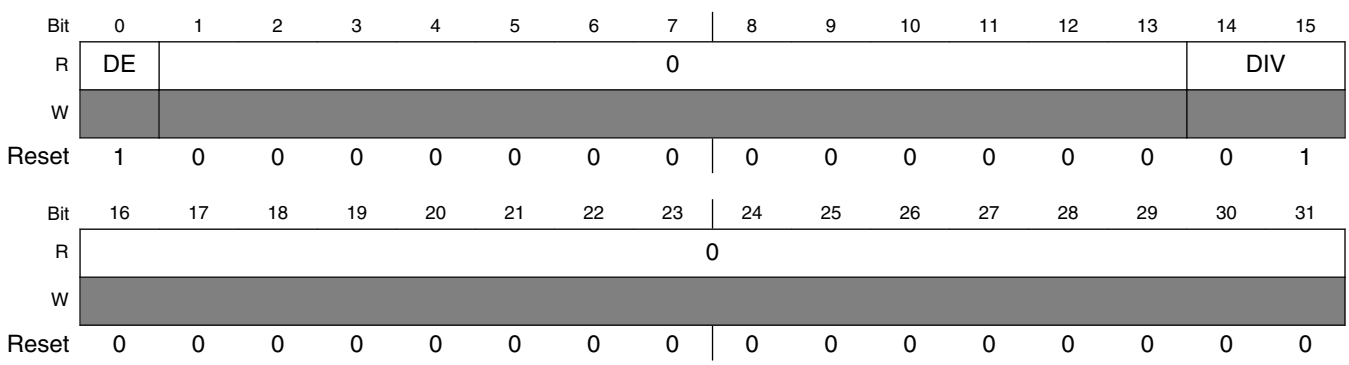
This register controls system clock divider 4 - F80 MHz peripheral clock.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

This register is reset only on POR reset, while on PHASE3 reset the clock divider loads the default reset value at the completion of division cycle corresponding to programmed value.

Address: 0h base + 7F8h offset = 7F8h



MC_CGM_SC_DC4 field descriptions

Field	Description
0 DE	Divider 4 Enable 1 Enable system clock divider 4
1–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 DIV	Divider 4 Division Value —The resultant F80 clock will have a period two times that of the system clock.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.28 System Clock Divider 5 Configuration Register (MC_CGM_SC_DC5)

This register controls system clock divider 5 - FS80 peripheral clock clock.

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access will not result in an exception, however the value will not be loaded with the new value.

This register is reset only on POR reset, while on PHASE3 reset the clock divider loads the default reset value at the completion of division cycle corresponding to programmed value.

Address: 0h base + 7FCh offset = 7FCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	0												DIV		
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_SC_DC5 field descriptions

Field	Description
0 DE	Divider 5 Enable 1 Enable system clock divider 5

Table continues on the next page...

MC_CGM_SC_DC5 field descriptions (continued)

Field	Description
1–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 DIV	Divider 5 Division Value —The resultant FS80 clock will have a period ‘DIV + 1’ times that of the system clock.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.29 Auxiliary Clock 2 Select Control Register (MC_CGM_AC2_SC)

This register is used to select the current clock source for the ENET0 and ENET1 timer clock.

This register is reset only on POR reset.

Address: 0h base + 840h offset = 840h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0							SELCTL	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC2_SC field descriptions

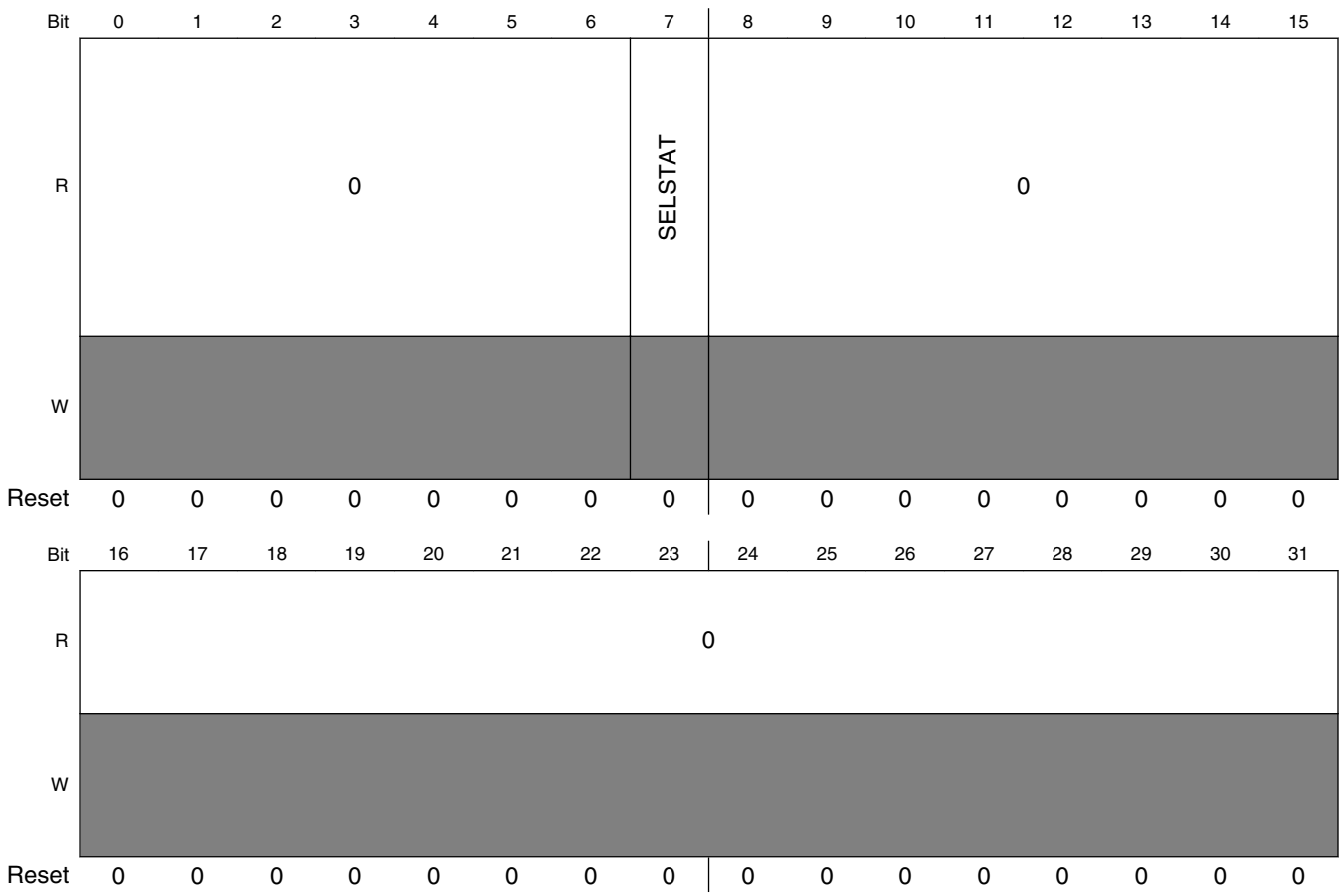
Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELCTL	Auxiliary Clock 2 Source Selection Control — This value selects the current source for auxiliary clock 2. 0 F40 clock 1 External clock
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.30 Auxiliary Clock 2 Select Status Register (MC_CGM_AC2_SS)

This register provides the current auxiliary clock 2 source selection

This register is reset only on POR reset.

Address: 0h base + 844h offset = 844h



MC_CGM_AC2_SS field descriptions

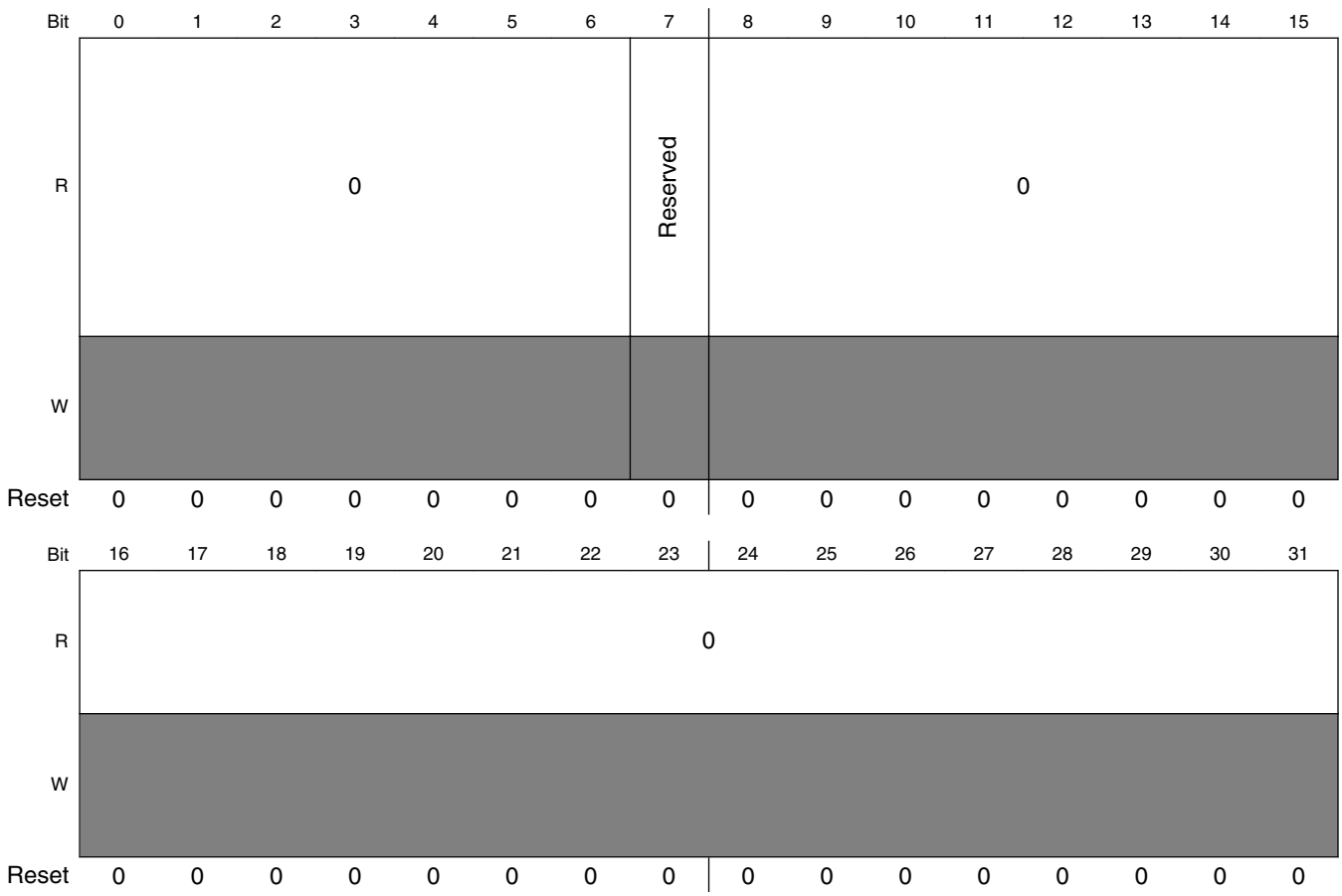
Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 SELSTAT	Auxiliary Clock 2 Source Selection Status — This value indicates the current source for auxiliary clock 2. 0 F40 clock 1 External clock
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.31 Auxiliary Clock 3 Select Status Register (MC_CGM_AC3_SS)

This register provides the current auxiliary clock 3 source selection.

This register is reset only on POR reset.

Address: 0h base + 864h offset = 864h



MC_CGM_AC3_SS field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.32 Auxiliary Clock 6 Select Control Register (MC_CGM_AC6_SC)

This register is used to select the current clock source for the CLKOUT_0.

This register is reset only on POR reset.

Address: 0h base + 8C0h offset = 8C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELCTL				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC6_SC field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELCTL	Auxiliary Clock 6 Source Selection Control — This value selects the current source for auxiliary clock 6. 0000 FXOSC 0001 FIRC 0010 PLL_CLKOUT1 0011 S160 0100 RTC clk 0101 PLL_CLOCKOUT2 0110 SXOSC 0111 SIRC 1000 Z4 CLK 1001 Reserved 1010 HSM Core CLK 1011 Reserved 1100 S40 1101 F40 1110 F80 1111 FS80
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.33 Auxiliary Clock 6 Select Status Register (MC_CGM_AC6_SS)

This register provides the current auxiliary clock 6 source selection for CLKOUT_0.

This register is reset only on POR reset.

Address: 0h base + 8C4h offset = 8C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				SELSTAT				0																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_CGM_AC6_SS field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 SELSTAT	Auxiliary Clock 6 Source Selection Status — This value indicates the current source for auxiliary clock 6. 0000 FXOSC 0001 FIRC 0010 PLL_CLKOUT1 0011 S160 0100 RTC clk 0101 PLL_CLOCKOUT2 0110 SXOSC 0111 SIRC 1000 Z4 CLK 1001 Reserved 1010 HSM Core CLK 1011 Reserved 1100 S40 1101 F40 1110 F80 1111 FS80
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

31.3.34 Aux Clock 6 Divider Configuration 0 for CLKOUT_0 (MC_CGM_AC6_DC0)

This register controls auxiliary clock 6 divider 0.

Access: User read, Supervisor read/write

NOTE

Byte and half-word write accesses are not allowed for this register. Such an access does not result in an exception, however the value is not loaded with the new value.

This register is reset only on POR reset, while on PHASE3 reset the clock divider loads the default reset value at the completion of division cycle corresponding to programmed value.

Address: 0h base + 8C8h offset = 8C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DE	Reserved										DIV				
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

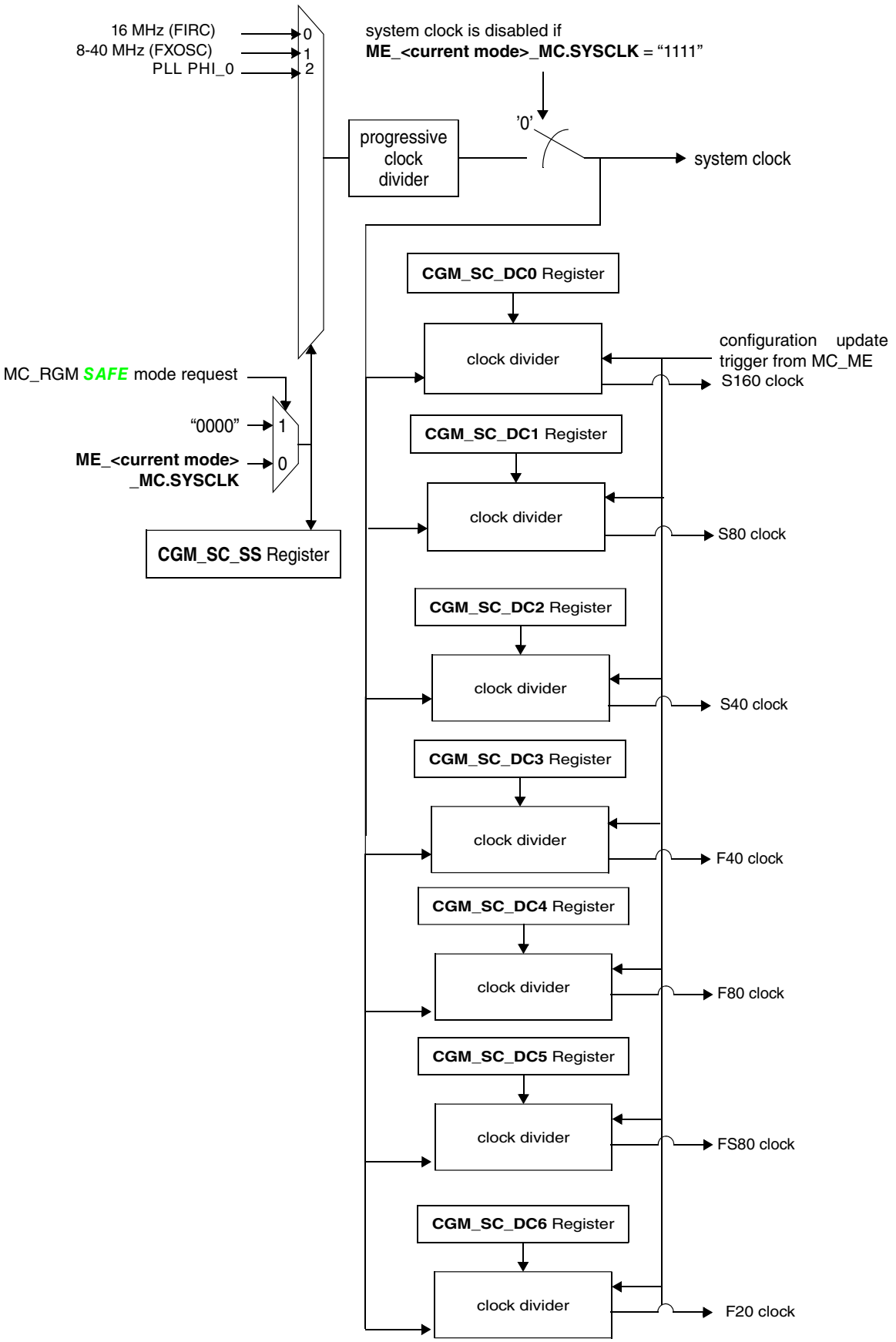
MC_CGM_AC6_DC0 field descriptions

Field	Description
0 DE	CLKOUT_0 Divider Enable 0 Disable auxiliary clock 6 divider 0 1 Enable auxiliary clock 6 divider 0
1–10 -	This field is reserved. Reserved
11–15 DIV	CLKOUT_0 Divider Division Value. The resultant CLKOUT0 pin clock has a period 'DIV + 1' times that of auxiliary clock 6. The DIV field is written to if and only if a) DE is 1 or b) 1 is written to DE.
16–31 -	This field is reserved. Reserved

31.4 Functional Description

31.4.1 System Clock Generation

The following figure shows the block diagram of the system clock generation logic. The MC_ME provides the system clock select and switch mask (see MC_ME chapter for more details), and the MC_RGM provides the safe clock request (see MC_RGM chapter for more details). The safe clock request forces the selector to select the 16 MHz internal RC oscillator as the system clock and to ignore the system clock select.



Note: MC_CGM_SC_DC1[DIV] and MC_CGM_SC_DC5[DIV] must be equal at all times.

Figure 31-2. MC_CGM System Clock Generation Overview

31.4.1.1 System Clock Source Selection

During normal operation, the system clock selection is controlled

- on a SAFE mode or reset event, by the MC_RGM via the safe_clock_req input
- otherwise, by the MC_ME via the clk_sys_src_sel input

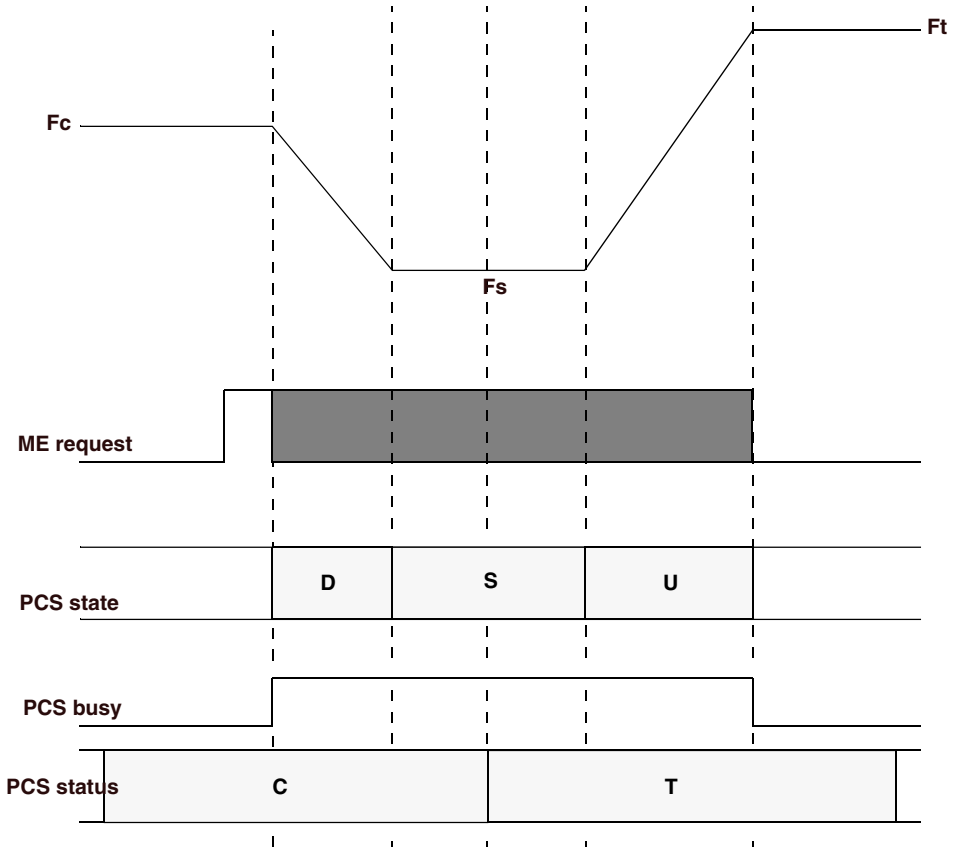
31.4.1.2 Progressive System Clock Switching

Progressive clock switching (PCS) is an option which simplifies the load changes when transitioning between modes by first going to a lower frequency, of the 16 MHz IRC, and then ramping the frequency with the new clock source selected.

The diagram below shows the frequency change in a device using PCS when the system moves from Fc to Ft. The current clock or Fc is gradually divided using the PCS configuration registers to reach 16 MHz which is the safe clock freq (Fs). Once the current clock frequency comes down to Fs, a clock switch to IRC is made. At this point all the peripherals and cores are switched on/off to match the target configuration of the MC_ME registers. Then the clock is switched from IRC to the target clock with dividers configured such that the target clock is running at 16 MHz (Fs). After that the dividers are changed to get a gradual ramp up to the target frequency(Ft), at this point, the PCS divider is dividing the target clock by 1.

NOTE

PCS configuration should not be done if some communication peripherals e.g., FlexCAN, LinFlex which are required to be working in a seamless manner during mode transition.



Note:
C: Current clock, T: Target clock, D: Clock Ramp down, S: Safe clock, U: Clock Ramp up

Figure 31-3. Progressive System Clock Switching

Based on the power level values of the current and target modes, the MC_ME requests the MC_CGM to ramp the system clock frequency down and/or up. During ramp-down, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVCn, and CGM_PCS_DIVEn registers, where n corresponds to the current system clock source selection. During ramp-up, the rate of the frequency change is based on the CGM_PCS_SDUR, CGM_PCS_DIVCn, and CGM_PCS_DIVSn registers, where n corresponds to the target system clock source selection.

It is recommended to program the PCS configuration registers (i.e. MC_CGM_PCS_SDUR, MC_CGM_PCS_DIVCn, MC_CGM_PCS_DIVEn and MC_CGM_DIVSn) with FIRC selected as the system clock. This ensures reliable progressive clock operation during the PCS register programming in case of a SAFE mode or RESET event.

31.4.1.2.1 Generic Clock Change Requirements

For a maximum allowed change of the frequency (f_{chg}) and for a given source clock frequency f_{src} (current or target clock source), the maximum allowed frequency change rate a_{max} is given in the following equation.

$$a_{max} = \frac{f_{chg}}{f_{src}}$$

Equation 6

31.4.1.2.2 Configuration of CGM_PCS_SDUR

The switch duration field CGM_PCS_SDUR[SDUR] defines the duration of one system clock switch step in terms of 16 MHz internal RC oscillator. After the expiration of this time, the module changes the clock divider value which chnages the frequency of the system clock.

31.4.1.2.3 Configuration of CGM_PCS_DIVC*n*[RATE]

For a given maximum allowed frequency change rate a_{max} the value d to be programmed into the PCS_DIVCn[RATE] register is given in the following table.

Table 31-1. MC_CGM CGM_PCS_DIVC*n*[RATE] Values

a_{max}	d = CGM_PCS_DIVC*n*[RATE]	CGM_PCS_DIVC*n*[RATE] in decimal	CGM_PCS_DIVC*n*[RATE] in hex
0.05	0.012	12	0x0C
0.10	0.048	48	0x30
0.15	0.112	112	0x70
0.20	0.184	184	0xB8

31.4.1.2.4 Generic Clock Change Properties

The number *k* of required steps to reach or leave the divided system clock with frequency f_{safe} from the source clock with frequency f_{src} is given in the following equation.

$$k = \left\lceil 0.5 + \sqrt{0.25 - \frac{2\left(1 - \frac{f_{src}}{16MHz}\right)}{d}} \right\rceil$$

Equation 7

31.4.1.2.5 Clock Ramp-Down

The clock ramp-down starts with the divider value 1 and with the given divider increment value PCD_DIVCn[RATE] and ends with the given divider value PCS_DIVEn[DIVE].

The divider end value for clock ramp-down is given in the following equation.

$$\text{PCS_DIVEn[DIVE]} = \frac{f_{curr}}{f_{safe}} \times 1000 - 1$$

Equation 8

Where f_{curr} is the frequency of the currently selected system clock source.

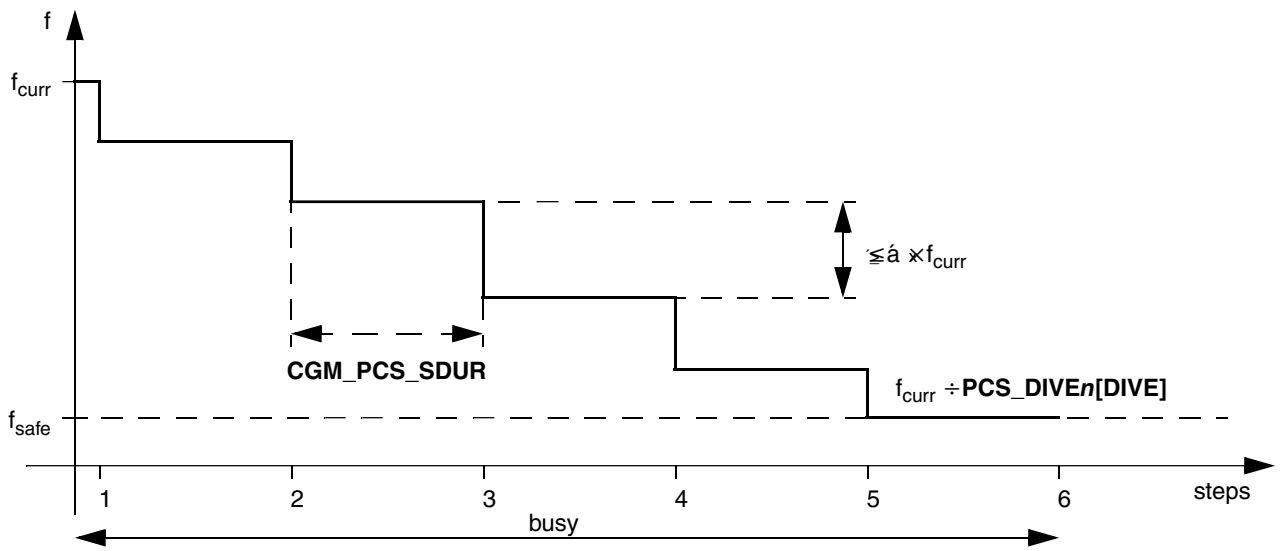


Figure 31-4. MC_CGM System Clock Ramp-Down Timing (k = 6 example)

31.4.1.2.6 Clock Ramp-Up

The clock ramp-up starts with the given divider value PCS_DIVSn[DIVS] and with the given divider decrement value PCD_DIVCn[INIT] and ends with the divider value 1.

The initial divider change start value PCS_DIVCn[INIT] for system clock ramp-up is given in the following equation.

$$\text{PCS_DIVCn[INIT]} = d \times k \times 1000$$

Where k is calculated as shown in the sectionGeneric Clock Change Properties using the target system clock source frequency f_{targ}. The divider start value for clock ramp-up is given in the following equation.

$$\text{PCS_DIVSn[DIVS]} = \left(1 + d\frac{k(k+1)}{2}\right) \times 1000 - 1$$

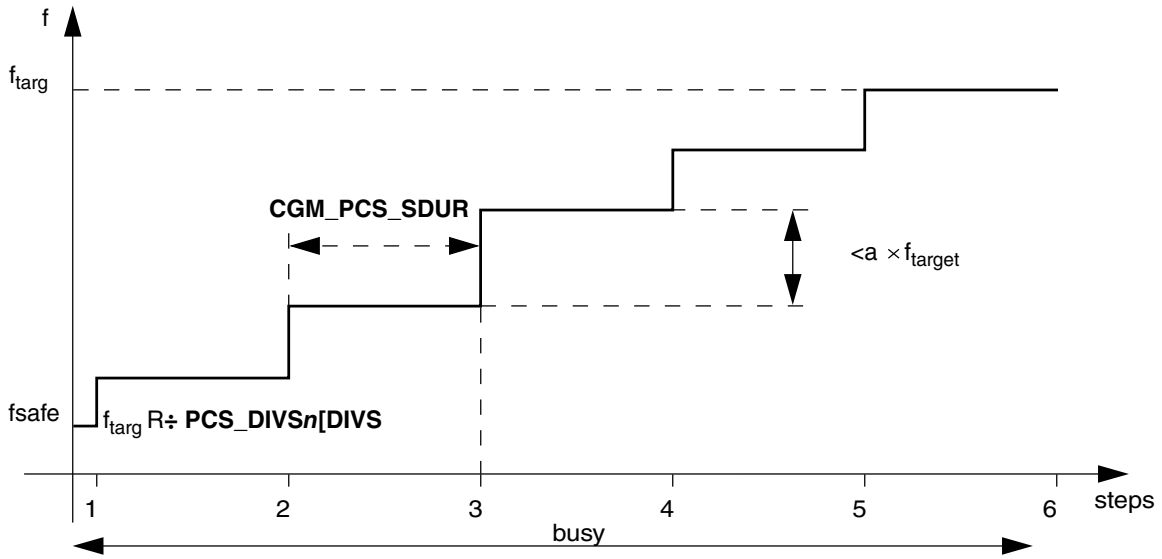


Figure 31-5. MC_CGM System Clock Ramp-Up Timing (k = 6 example)

31.4.1.3 System Clock Disable

During the TEST mode, the system clock can be disabled by the MC_ME.

31.4.1.4 System Clock Dividers

The MC_CGM generates the following derived clocks from the system clock:

- S160 clock - controlled by the CGM_SC_DC0 register
- S80 clock - controlled by the CGM_SC_DC1 register (selected modules - Flexray RAM and DMA_RAM).
- S40 clock - controlled by the CGM_SC_DC2 register
- F40 peripheral clock - controlled by the CGM_SC_DC3 register
- F80 peripheral clock - controlled by the CGM_SC_DC4 register
- FS80 clock - controlled by the CGM_SC_DC5 register
- F20 clock - controlled by the CGM_SC_DC6 register

31.4.2 Auxiliary Clock Generation

The following figures show the block diagrams of the generation logic for the various auxiliary clocks. See the following sections for auxiliary clock selection control:

- Auxiliary Clock 2 Select Control Register (MC_CGM_AC2_SC)

-
-
- Auxiliary Clock 5 Select Control Register (MC_CGM_AC5_SC)
- Auxiliary Clock 6 Select Control Register (MC_CGM_AC6_SC)

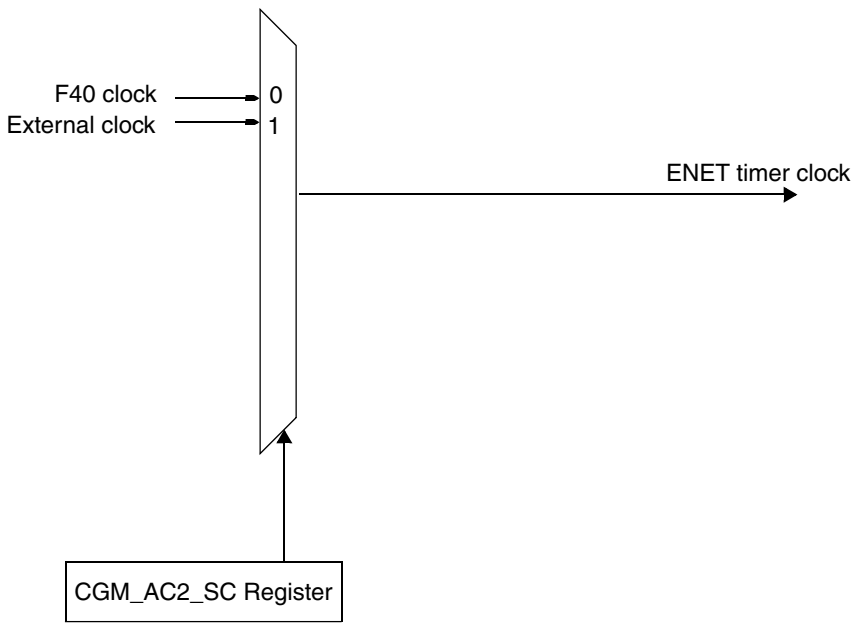


Figure 31-6. MC_CGM Auxiliary Clock 2 Generation Overview

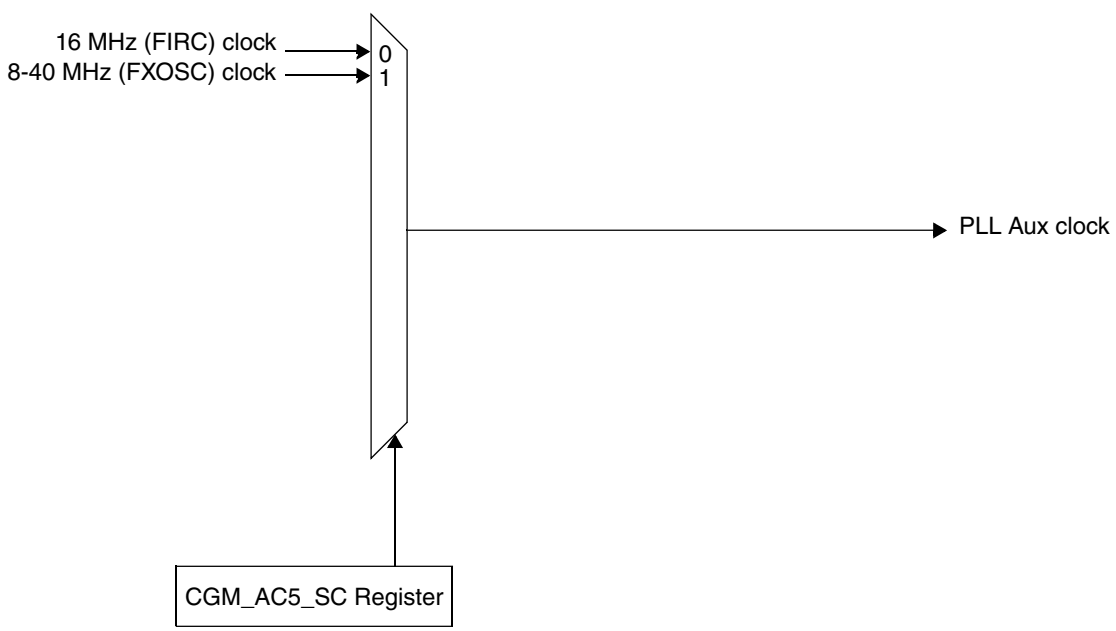


Figure 31-7. MC_CGM Auxiliary Clock 5 Generation Overview

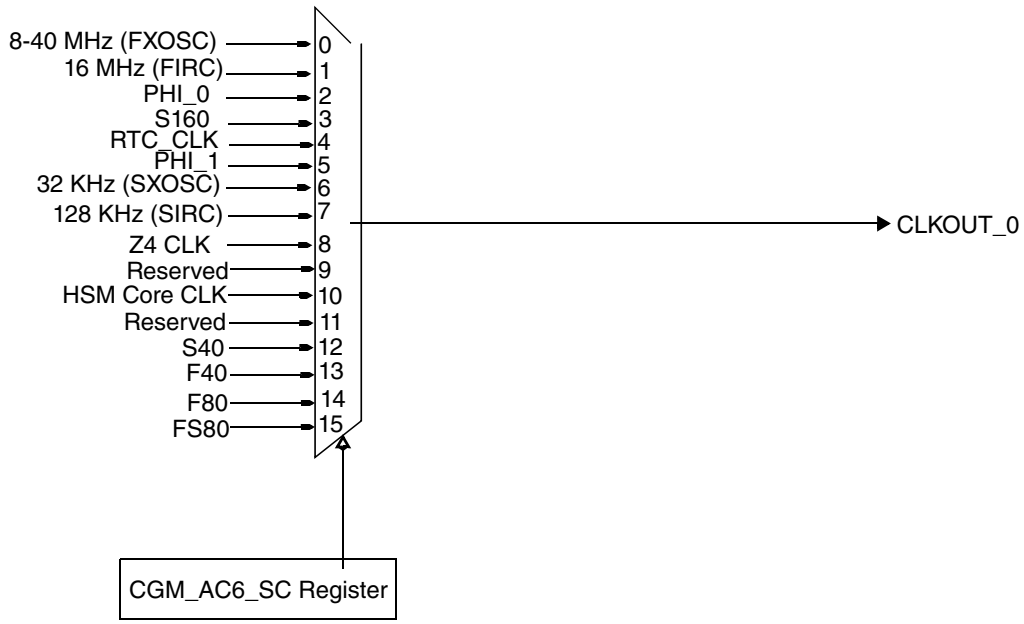


Figure 31-8. MC_CGM Auxiliary Clock 6 Generation Overview

31.4.3 Dividers Functional Description

Dividers are used for the generation of divided system and peripheral clocks. The MC_CGM has the following control registers for built-in dividers:

- System Clock Divider 0 Configuration Register (MC_CGM_SC_DC0)
- System Clock Divider 2 Configuration Register (MC_CGM_SC_DC2)
- System Clock Divider 3 Configuration Register (MC_CGM_SC_DC3)
- System Clock Divider 4 Configuration Register (MC_CGM_SC_DC4)
- System Clock Divider 5 Configuration Register (MC_CGM_SC_DC5)

The reset value of all counters is '1'. If a divider has its DE bit in the respective configuration register set to '0' (the divider is disabled), any value in its DIV field is ignored.

31.4.4 Halt Abort

A divider update trigger with SYS_DIV_RATIO_CHNG bit set will assert a request to the crossbar switch to halt the ongoing transactions. The dividers are updated once the crossbar switch acknowledges the request for halt. This protocol maybe aborted by urgent

system interrupts. The time instant at which the abort is asserted determines if the crossbar halt is aborted or the abort is ignored. If the abort request comes before the halt is acknowledged form the crossbar, the halt handshake is aborted.

The CGM_DIV_UPD_ABRT.DIV_UPD_ABRTD bit is set after a successful abort and must be read by software to take necessary steps. In case the CGM_DIV_UPD_ABRT.DIV_UPD_ABRTD bit is set, the software should initiate the divider update trigger at some later point. Till such a time that the dividers are updated again, the current division value of the dividers would not match the divider configuration. The CGM_DIV_UPD_STAT bit is de asserted after the abort is successfully completed signalling that no divider update is ongoing.

The halt abort feature may be disabled by software by setting the CGM_SC_DIV_RC.HALT_ABORT_DIS CGM_HALT_ABORT_DIS.

Chapter 32
Input Glitch Filter

32.1 Introduction

The Input Glitch Filter provides programmable glitch rejection filtering of digital input signals.

32.2 Block Diagram

The figure below is the block diagram for the Input Glitch Filter module.

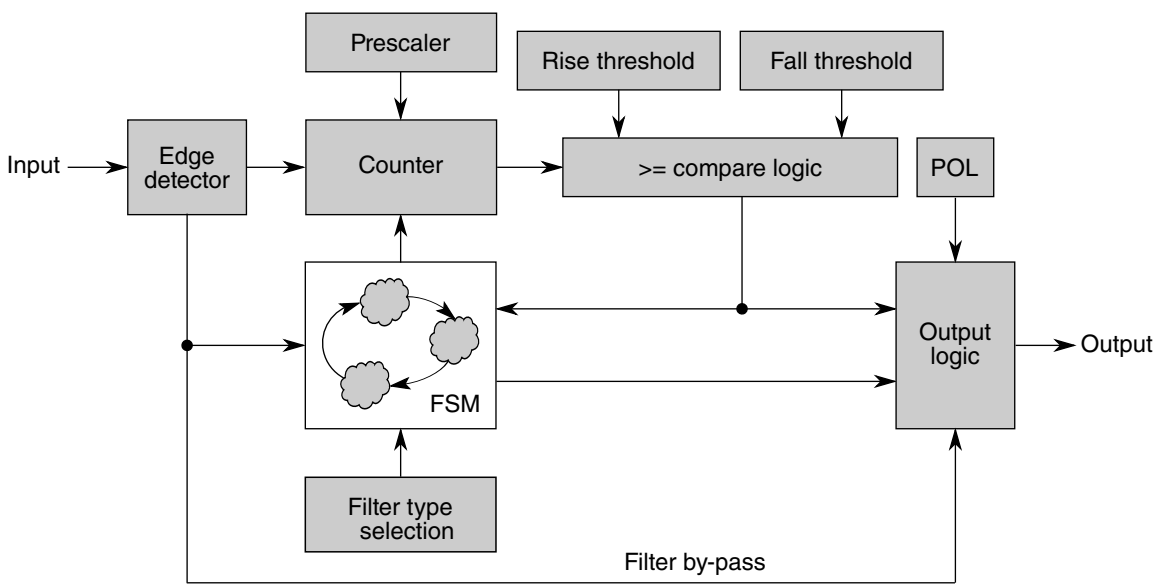


Figure 32-1. Input Glitch Filter block diagram

32.3 Input Glitch Filter overview

The Input Glitch Filter receives an input signal usually from a device PAD. Signals generated externally to the device are affected by environmental noise or glitches. The signal level has to be evaluated with some criteria in order to remove the noise which could affect the measurements done over this signal, such as transition time, period and level.

In a typical application, the Input Glitch Filter interfaces with one or more timer peripherals.

The strategy to remove glitches from an input signal is to monitor the level of the signal during a wide time window thus the glitches can be evaluated and removed.

32.3.1 Interaction with other peripherals

The figure below shows an example of how the Input Glitch Filter interacts with on-chip peripherals.

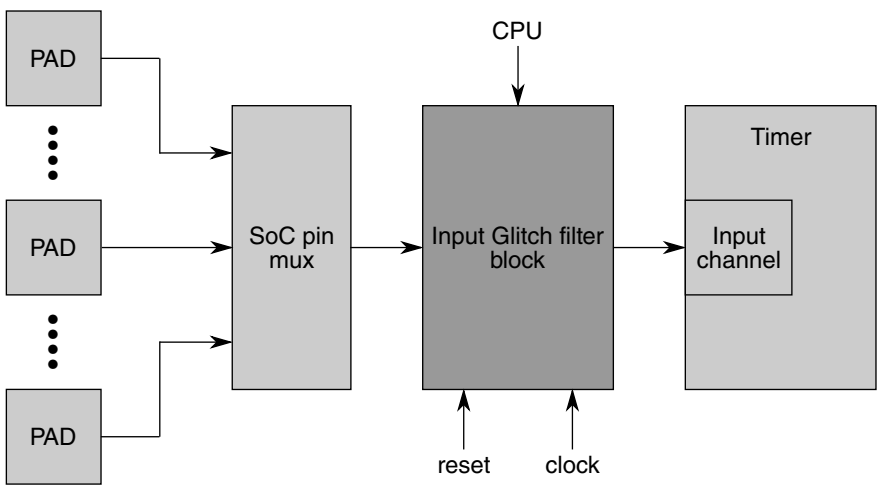


Figure 32-2. Input Glitch Filter application example

32.3.2 Mode of operation

The Input Glitch Filter has the following modes of operation:

- Configuration mode
- Normal mode

- Debug mode
- Low power mode

32.3.2.1 Configuration mode

The configuration mode is the initial mode of the Input Glitch filter after system reset is applied. The configuration mode is selected when FGEN = ‘0’ in the MCR register. The filter parameters such as rising and falling thresholds can be programmed when the filter is in configuration mode without disturbing the internal device logic since the filter output is kept at logic '0' state. Once FGEN is programmed with logic ‘1’ the filter enters the normal operating mode and is able to initiate the filtering process.

NOTE

The MDIS bit in the MCR register, if set to logic '1', makes the filter transition to low power mode independent of FGEN bit settings. Thus, in order to enter the configuration mode MDIS bit needs to be at logic '0'.

32.3.2.2 Normal mode

The normal mode of operation is entered when MCR register FGEN = ‘1’. In this mode the filter implements one of the filter types:

- Bypass filter
- Windowing filter
- Windowing filter with post-sample
- Integrating filter
- Integrating-hold filter

These filter types are described in the following sections. Before entering the normal mode, the filter parameters, such as rise and fall thresholds, need to be programmed in the appropriate registers, as well as the type needs to be defined in the corresponding register.

NOTE

In the normal mode it is still possible to change the filter parameters, such as rise and fall thresholds. See section On-the-fly filter reconfiguration for more details.

32.3.2.3 Debug mode

The debug mode is entered when the MCR FRZ bit is set and a system level debug request occurs. Once the debug mode is entered, the internal filter counter is halted and the debug functions controlled by the FBP, FOH, and FOL bits are enabled. The filter output is frozen and remains at the state it was immediately before the debug mode was entered. The FOH and FOL MCR bits can be used to force the filter output to logic '1' or '0', respectively. These bits are self-negating, thus they always read as zero.

By setting the MCR FBP bit, the filter input state is copied to the filter output, thus allowing to control the filter output based on the filter input state. The output is forced during one clock cycle. After that the filter output remains at the new forced state until it is changed again by using a debug control bit or if the filter returns to the normal mode of operation.

NOTE

The system level debug request occurs when a specific event, or a series of events, occurs in one of the device sub-modules and is interpreted by the device debug logic as a debug request. The Input Glitch Filter is not capable of generating an event that causes the assertion of the system level debug request.

32.3.2.4 Low power mode

The low power mode is controlled by the MDIS bit in the MCR register or by the STOP signal which is the system level low power control signal. When entering low power mode the filter output remains unchanged at the state it was just before the low power mode was entered. Since the Input Glitch Filter clocks are stopped in this mode, the internal counter remains inactive. The description of this mode is covered in the Low Power Mode section of this document.

32.4 Memory map

The Input Glitch Filter registers are implemented in 32-bit. 8-bit write operations are allowed. The coherency in case of 8-bit access must be provided by the application software since there is no coherence mechanism built into the hardware.

NOTE

RESERVED bit registers, when available for write operations, should be written always with logic '0'. Read access does not return meaningful data from RESERVED bit registers.

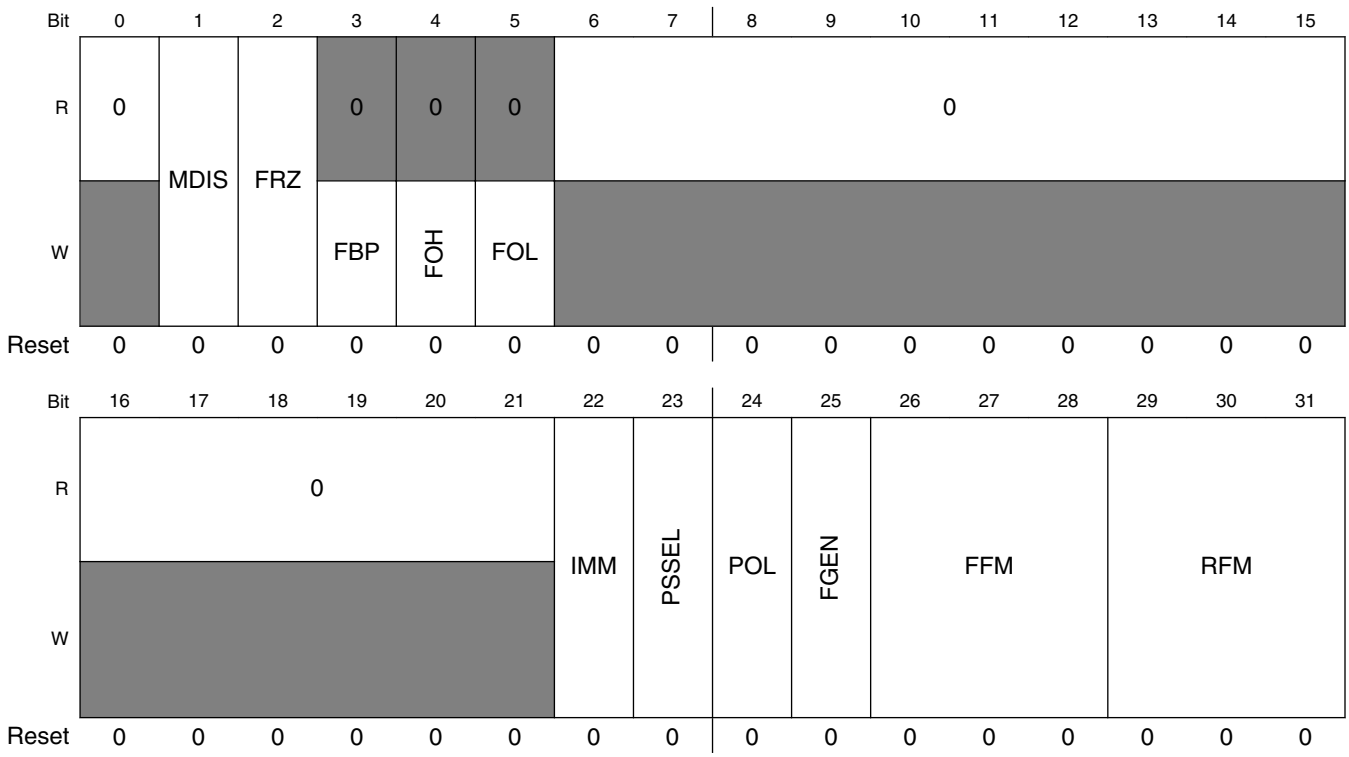
IGF memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (IGF_MCR0)	32	R/W	0000_0000h	32.4.1/684
4	Module Status Register (IGF_MSR0)	32	R/W	0000_0000h	32.4.2/687
10	Prescaler Register (IGF_PRESR0)	32	R/W	0000_0000h	32.4.3/688
1C	Rising edge Threshold Register (IGF_RTHR0)	32	R/W	0000_0000h	32.4.4/689
20	Falling edge Threshold Register (IGF_FTHR)	32	R/W	0000_0000h	32.4.5/689

32.4.1 Module Configuration Register (IGF_MCR*n*)

The Module Configuration register (MCR) provides access to low power, debug, and filter control bits.

Address: 0h base + 0h offset + (0d × i), where i=0d to 0d



IGF_MCR*n* field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 MDIS	Module Disable The MDIS bit forces the filter to enter a low power state. After this bit is asserted the clock for the module is removed thus filter operation is not possible. The access to filter registers is allowed. 0 Input Glitch Filter is not in low power mode due to MDIS bit 1 Input Glitch Filter is in low power mode
2 FRZ	Freeze bit for debug operation The FRZ bit enables the Input Glitch Filter to enter freeze mode (debug mode) when a system level signal request the module enters freeze (debug) mode. Once in freeze mode the filter operation is disabled meaning that transitions on the input signal will not propagate through the filter output. The bypass FBP bit can be used by the software to force the filter output to be the same as the filter input. In this mode FOH and FOL also can be used to define the filter output state.

Table continues on the next page...

IGF_MCR*n* field descriptions (continued)

Field	Description
	0 Freeze mode disabled 1 Freeze mode enabled
3 FBP	Force Bypass This bit is self-negated thus is only read as zero. Writing 1 to this bit while in freeze mode forces the filter to bypass the value in the input to the filter output, thus allowing a software control on the filter. This bit is functional only in freeze mode. 0 no filter bypass 1 transfer input signal value to the filter output
4 FOH	Force filter output high If set, this bit forces the Input Glitch Filter output to '1'. This bit is functional only in debug mode. It is self-negated thus always read as zero. 0 no action 1 forces the filter output to '1'
5 FOL	Force filter output low If set this bit forces the Input Glitch Filter output to '0'. This bit is functional only in debug mode. It is self-negated thus always read as zero. 0 no action 1 forces the filter output to '0'
6–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 IMM	Immediate edge propagation control bit The IMM bit controls the propagation of an edge through the filter. If asserted, this bit defines that an edge at the filter input propagates through the filter output independent of the prescaler settings. In bypass mode the propagation delay is three system clock cycles. If negated, the signal propagation depends upon the prescaler settings. The filter hardware will only allow chages to this bit if the filter is disabled, FGEN = 0 0 edge propagation depends on prescaler 1 edge propagation within three system clock cycles
23 PSSEL	Prescaler selection bit The PSSEL selects if the prescaler used by the Input Glitch Filter is from internal source (internal prescaler counter) or from external source. The filter hardware will only allow changes to this bit if the filter is disabled, FGEN=0. 0 internal prescaler selected 1 external prescaler selected
24 POL	Output polarity bit This bit sets the polarity of the filter output to inverted or not inverted. The polarity control implemented by the POL bit is applied at the filter output stage thus also changing the reset state of the output, even if the filter is disabled by the FGEN bit. The filter hardware will only allow changes to this bit if the filter is disabled, FGEN=0.

Table continues on the next page...

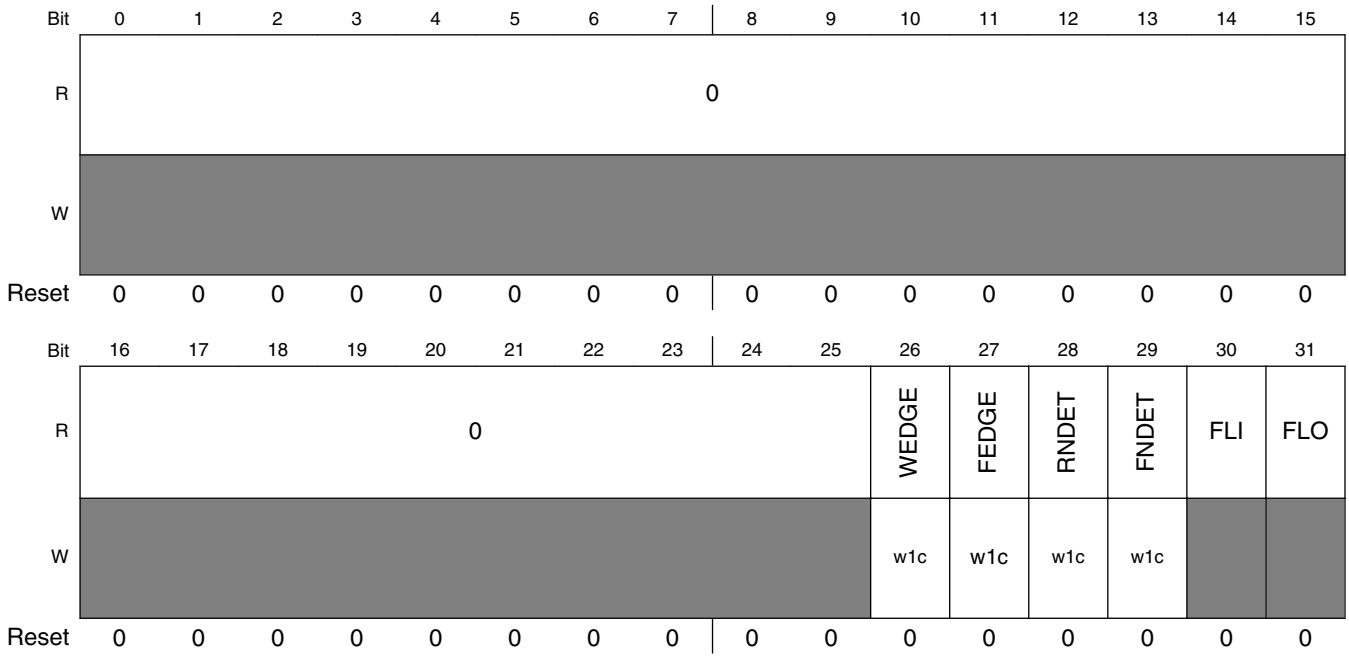
IGF_MCR*n* field descriptions (continued)

Field	Description
	0 not inverted output 1 inverted output
25 FGEN	Filter global enable This control bit enables the filter operation, meaning that if the filter is enabled the filtering selected types are applied to the rising and falling edges of the input signal. If the filter is not enabled, then the filter output remains unchanged independent of the filter input signal. Note that the filter output state is affected by the POL bit when the filter is disabled. 0 filter is disabled: filter output keeps current state. 1 filter is enabled: filtering is applied to selected edges.
26–28 FFM	Falling edge filter type selection This bit field selects the filter type for the filter input signal falling edge. The filter hardware will only allow changes to this bit if the filter is disabled, FGEN=0. 000 bypass: edge is propagated to filter output without any filtering in three system clock cycles 001 windowing: windowing filter is selected for the falling edge 010 integrating: integrating filter is selected for the falling edge 011 integrating-hold: integrating-hold filter is selected for the falling edge 100 windowing with post sample: windowing filter with post sample is selected for the falling edge
29–31 RFM	Rising edge filter type selection This bit field selects the filter type for the input signal rising edge. The filter hardware will only allow changes to this bit if the filter is disabled, FGEN=0. 000 bypass: edge is propagated to filter output without any filtering in three system clock cycles 001 windowing: windowing filter is selected for the rising edge 010 integrating: integrating filter is selected for the rising edge 011 integrating-hold: integrating-hold filter is selected for the rising edge 100 windowing with post sample: windowing filter with post sample is selected for the rising edge

32.4.2 Module Status Register (IGF_MSR*n*)

The Module status register (MSR) provides access to filter status bits and corresponding clear bits as well as access to the state of filter input and output signals.

Address: 0h base + 4h offset + (64d × i), where i=0d to 0d



IGF_MSR*n* field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 WEDGE	Filter is active waiting for an edge This bit is set when the filter is enabled but the internal counter is not active. It indicates that the filter is waiting for an edge to occur at the filter input. It is set by the Input Glitch Filter hardware and cleared by the software by writing 1 to the bit. Since the intention of this bit is to capture the dynamic of the filter hardware, it should be cleared after the read thus being able to capture a new event. 0 Input Glitch Filter is disabled or the internal counter is active 1 Input Glitch Filter is waiting for an edge
27 FEDGE	Filter is active processing an edge This bit is set during the filter active state thus when the filter internal counter is counting, processing an edge in a filtering mode. It is cleared by the software by writing 1 to the bit. Since the intention of this bit is to capture the dynamic of the filter hardware, it should be cleared after the read thus being able to capture a new event. 0 Input Glitch Filter internal counter is not active 1 Input Glitch Filter internal counter is active

Table continues on the next page...

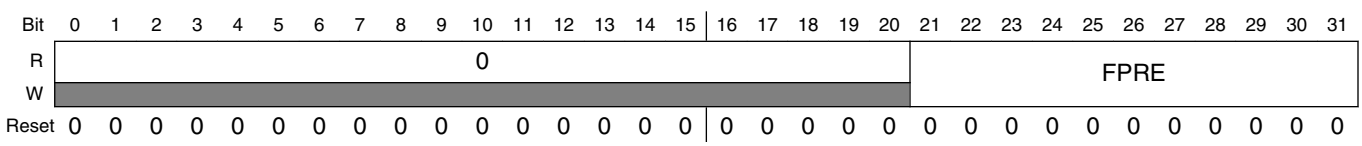
IGF_MSR*n* field descriptions (continued)

Field	Description
28 RNDET	Rise noise detected bit This bit indicates that the filter filtered out noise in the input signal after a rising edge was detected. The criteria to set this bit is when a rising edge is detected thus starting the filter counter and before this edge pass through the filter (if filter is in integrating filter types) or during a window (if filter is in windowing filter types) another edge occurs in the input signal. This bit is set by a noise detection and cleared by the software by writing 1 to this bit. 0 noise was not detected 1 noise was detected after a rising edge
29 FNDET	Fall noise detected bit This bit indicates that the filter filtered out noise in the input signal after a falling edge was detected. The criteria to set this bit is when a falling edge is detected, thus starting the filter counter, and, if the filter is in integrating types, before this edge passes through the filter, or, in windowing types, when another edge occurs in the input signal during a window. This bit is set by a noise detection and cleared by the software by writing 1 to this bit. 0 noise was not detected 1 noise was detected after a falling edge
30 FLI	Filter input The FLI bit follows the state of the filter input value as captured by the filter input logic, which is affected by the IMM bit and prescaler settings.
31 FLO	Filter output The FLO bit follows the state of the filter output value considering the POL bit state.

32.4.3 Prescaler Register (IGF_PRESR*n*)

The prescaler register (PRESR) provides access to the prescaler bits.

Address: 0h base + 10h offset + (64d × i), where i=0d to 0d



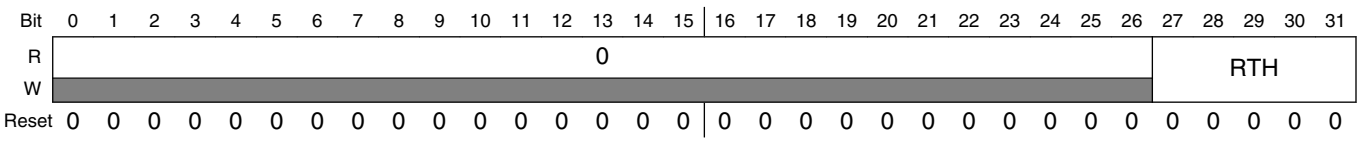
IGF_PRESR*n* field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–31 FPRE	Filter prescaler The prescaler defines the rate of the filter counter FCOUNT. The filter counter clock rate is defined by the formula: Filter counter clock = module clock / (FPRE + 1).

32.4.4 Rising edge Threshold Register (IGF_RTHR*n*)

The Rising edge Threshold register (RTHR) provides access to the rising edge threshold bits.

Address: 0h base + 1Ch offset + (64d × i), where i=0d to 0d



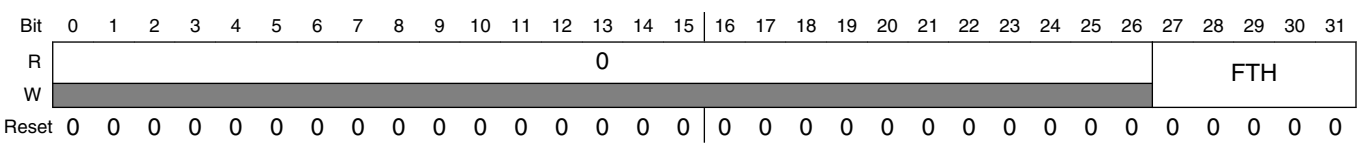
IGF_RTHR*n* field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 RTH	Rising edge threshold This bit field defines the filter counter threshold when a rising edge occurs is being filtered.

32.4.5 Falling edge Threshold Register (IGF_FTHR)

The Falling edge Threshold register (FTHR) provides access to the falling edge threshold bits.

Address: 0h base + 20h offset = 20h



IGF_FTHR field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 FTH	Falling edge threshold This bit field defines the filter counter threshold when a falling edge is being filtered.

32.5 Functional description

The following sections describe the Input Glitch Filter module functionality.

32.5.1 Input double sampling

The filter input is double sampled for clock domain translation thus avoiding asynchronous signals to propagate through the filter internal logic. This stage is composed by two flip-flops connected back to back. The delay caused by this logic is therefore two clock cycles.

Note that due to the synchronous nature of the Input Glitch Filter design, if a glitch smaller than one clock cycle occurs at the filter input it will not propagate to the filter internal logic and thus will not propagate to the filter output. Also, if clock is not applied to the filter (i.e. in low power mode) any transition at the filter input will not propagate to the filter output.

32.5.2 Filter types

The Input Glitch Filter implements four main filter types:

- Windowing
- Windowing with post sample
- Integrating
- Integrating hold

These filter types can be selected for the rising and falling edges independently, thus allowing different filter types to be selected for different edges (i.e. rising edge filtered with windowing filter and falling edge filtered with integrating).

32.5.2.1 Windowing filter

This filter type detects an edge (positive or negative edge) at the filter input and then mask out detection of subsequent edges over a period of time, which is defined as the blanking period, thus effectively behaving as a high pass filter. After an edge is detected the level at the filter input is propagated to the filter output within three clock cycles. The delay is caused by the synchronization stage at the input and by the filter output register stage. During the blanking period any transition at the filter input is disregarded. After the blanking period has elapsed, the filter compares the input level state with the filter output state. In case they are different a transition is inferred and re-evaluate according to the filter type assigned to that transition (positive or negative edges). Since, in this example,

the windowing filter type is programed for both edges, the inferred negative edge is propagated to the filter output and another blanking period is initiated. The figure below describes the windowing filter type behavior.

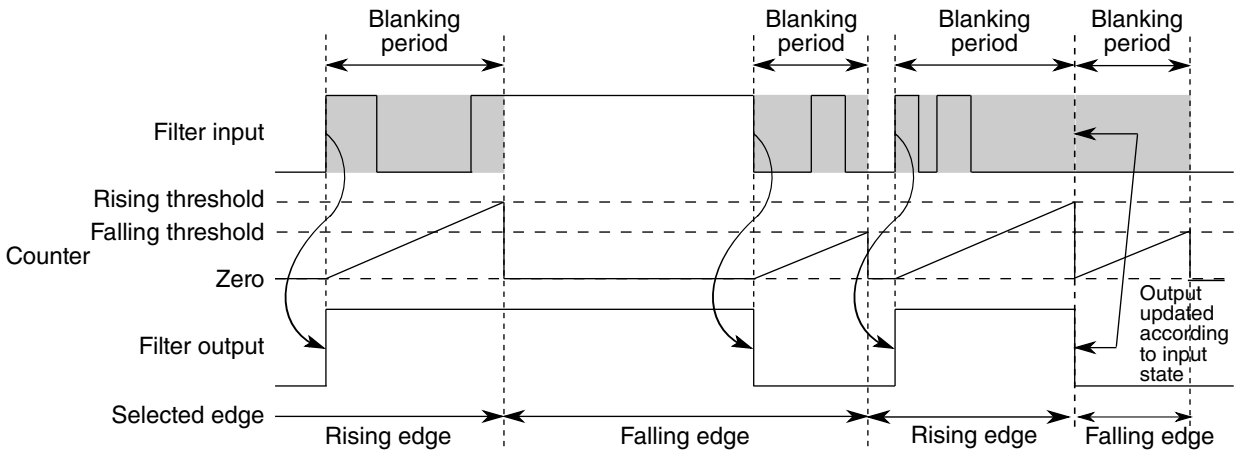


Figure 32-3. Windowing filter type

32.5.2.2 Windowing filter with post sample

This filter type detects an edge (positive or negative edge) at the filter input and then masks out detection of subsequent edges over a period of time, which is defined as the blanking period. During the blanking period, any transition at the filter input is disregarded.

When the blanking period finishes, the filter input level is propagated to the filter output. If the filter input is equal to the filter output when the blanking period finishes, then the filter output is not changed at the end of blanking period.

The figure below describes the windowing filter with post sample.

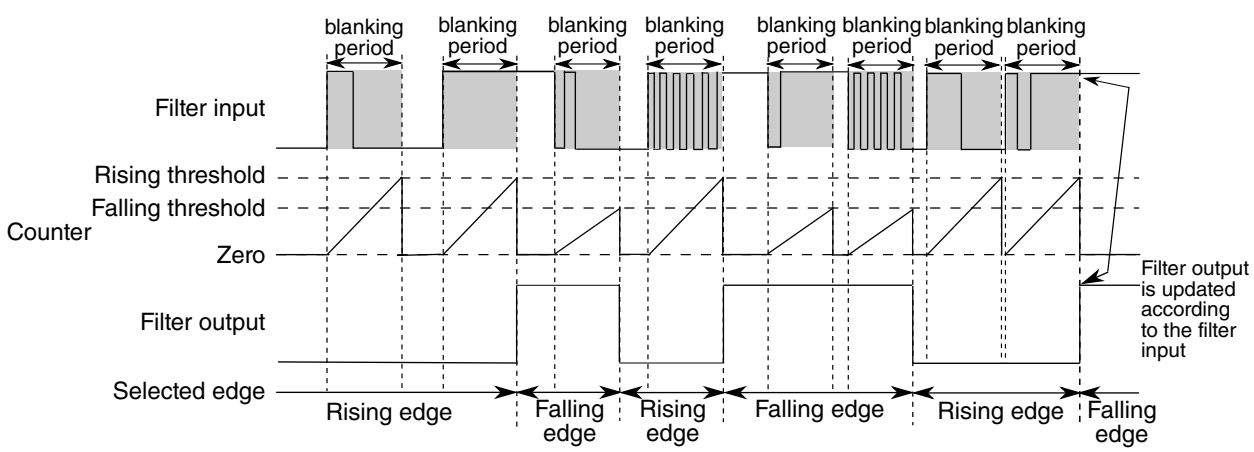


Figure 32-4. Windowing filter with post sample type

The windowing filter with post sample type is used to remove glitches and sparks around an edge.

32.5.2.3 Integrating filter

The integrating filter behaves like a low-pass filter, thus removing high frequency bursts from the input signal. The input and output signals are compared and if they are at different states an internal filter counter is enabled.

The direction of the counter (up or down counting) is defined by the states of the input and output filter signals as follows:

- Filter output is at the same state as the filter input: down-counting
- Filter output is at different state as the filter input: up-counting

The counter is compared with the rising or falling threshold register according to:

- Rising threshold register is selected if the rising edge is being filtered.
- Falling threshold register is selected if the falling edge is being filtered.

The filter output changes state to the same state as the filter input:

- If filter counter reaches the selected threshold for the edge being filtered.

The internal counter is disabled and cleared:

- If filter counter reaches the selected threshold for the edge being filtered.

The figure below describes an example of the Integrating filter type operation.

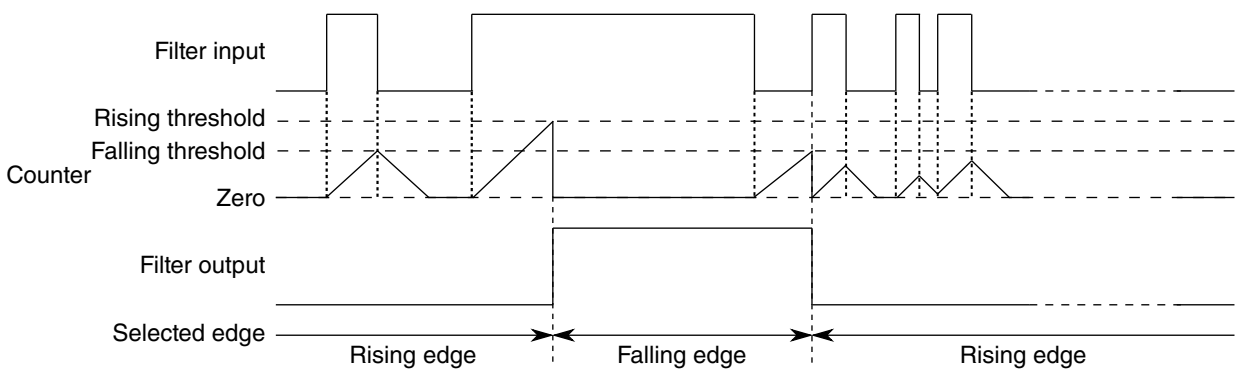


Figure 32-5. Integrating filter type

32.5.2.4 Integrating-hold filter

The integrating-hold filter type is similar to the integrating filter. The difference is the counter, which only counts in up direction instead of up-down in the integrating mode. In this filter configuration the input and output signals are compared and if they are at different states the filter counter is enabled. This counter is controlled by the state of the filter input and filter output signals as follows:

- Filter output is at the same state as the filter input: counter is in hold state (no-counting).
- Filter output is at different state as the filter input: counter is counting up.

The counter is compared with the rising or falling threshold register according to:

- Rising threshold register is selected if the rising edge is being filtered.
- Falling threshold register is selected if the falling edge is being filtered.

The filter output changes state to the same state as the filter input:

- If the filter counter reaches the selected threshold for the edge being filtered.

The internal counter is disabled and cleared:

- If the filter counter reaches the selected threshold for the edge being filtered.

The figure below shows an example of the Integrating-hold filter type operation.

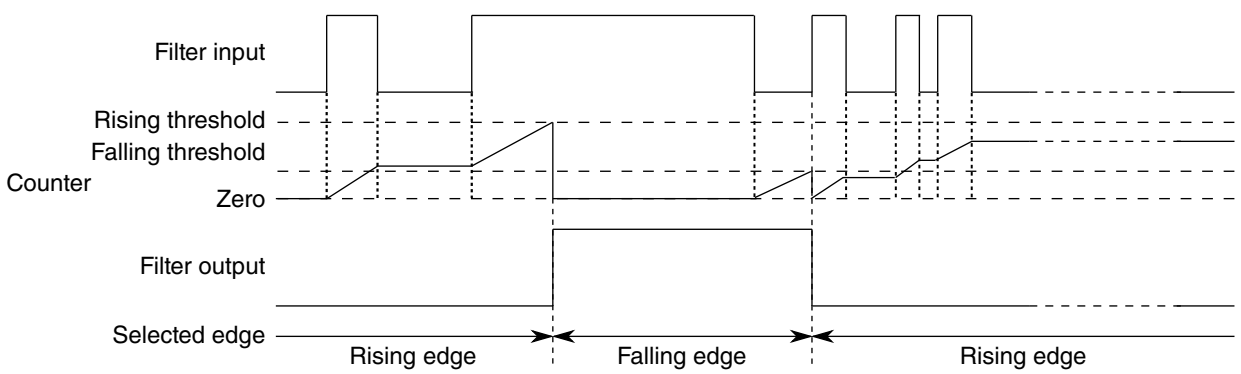


Figure 32-6. Integrating-hold filter type

32.5.2.5 Bypass filter

In the bypass filter type, the Input Glitch Filter immediately propagates any transition on the filter input to the filter output. This filter type does not provide any protection against glitches in the input signal. It is intended to be used in situations where the input signal is already filtered, or it is a glitch free signal, or the filter output is connected to a module that already has filtering capabilities. The bypass filter type is independently selected for rising or falling edges. Thus rising edges may be not filtered by the bypass filter type while the falling edges are filtered by any of the available filter types.

The bypass filter presents a delay in the edge propagation through the filter that is caused by the double sampling at the filter input and by the output filter register. Therefore, the total delay is three clock cycles.

32.5.2.6 Combined filter

The Input Glitch Filter allows the combination of filter types for different edges, positive and negative. As an example, the rising edges may be filtered using the windowing filter while the falling edges are filtered with an integrating filter. This configuration of the Input Glitch Filter is provided for situations where the noise present in the rising edges is different from the noise in the falling edges. Another use case is when the delay requirements of the input signal is different for rising and falling edges.

The figure below shows the combined filter operation. The windowing filter type is applied to the rising edges. The integrating filter is used for the falling edges.

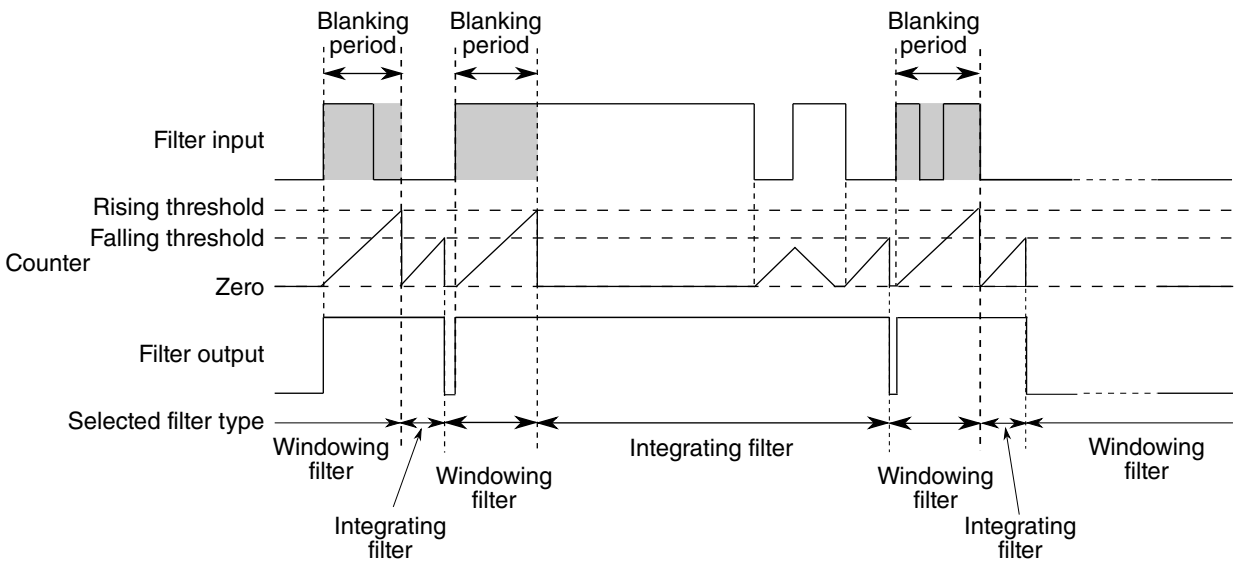


Figure 32-7. Combined filter type (windowing + integrating)

32.5.3 Application

This section describes several applications for the Input Glitch filter module.

32.5.3.1 Application use case 1: integrating filter

The integrating filter type is particularly useful in applications where glitches occur spuriously while the signal is expected to be in a steady state (either 1 or 0). Choosing correct up and down filter clock rates can ensure that the effects of spurious glitches can be completely eliminated, whilst minimizing propagation delays.

The application must be able to tolerate delays between the actual edge occurrence and the time that the signal is propagated through the filter logic. Because the delays are constants defined by the detection thresholds, application software may compensate for them, provided little or no noise occurs on the detected edge.

32.5.3.2 Application use case 2: windowing filter

The windowing filter type is particularly useful for removing noise bounce produced by a switch or sensor when its signal changes state. In this case, there is no propagation delay (except for the filter input synchronization logic and output register) between the filter input and filter output signals. This type of filter would not necessarily provide much immunity to spurious noise that occurred during the steady state portion of the signal. In order to optimize the characteristics of this filter, the blanking period should be of the same magnitude as the expected input signal rate. Signals that change frequency can be accommodated by software by dynamically updating the blanking period.

32.5.3.3 Application use case 3: windowing filter

Safety applications may need to compare the inputs obtained from different input pins, but derived from a single external source. This form of redundancy may detect non-common mode faults that may occur in the portions of the SoC that route and process the incoming signal. Having two different instances of windowing filter used on each input pin will ensure that the signal edge timing for both input paths is equal. Applying different filter characteristics for each input path can enhance the ability to detect SoC fault conditions.

32.5.3.4 Application use case 4: combined filter

Some applications may have rising edge signals that exhibit different characteristics to falling edge signals which might require different filtering to optimize the filtering effect. For example the “first” rising edge may represent a true real time event with a fast slew rate that must be captured without delay, while the falling edge signal may have a slow slew rate with many noisy transitions. The Input Glitch Filter can apply, for example, a

windowing filter on the rising edge to ensure no loss of real time information and an integrating filter on the falling edge to eliminate noise from being detected as a new rising transition.

32.5.3.5 Application use case 5: prescaler settings in windowing filter

The prescaler settings influence the timing for an edge to propagate through the filter. In windowing filter the IMM bit in the MCR register controls the way an edge is propagated through the filter. The figure below describes the windowing filter operation with regard to the IMM bit settings.

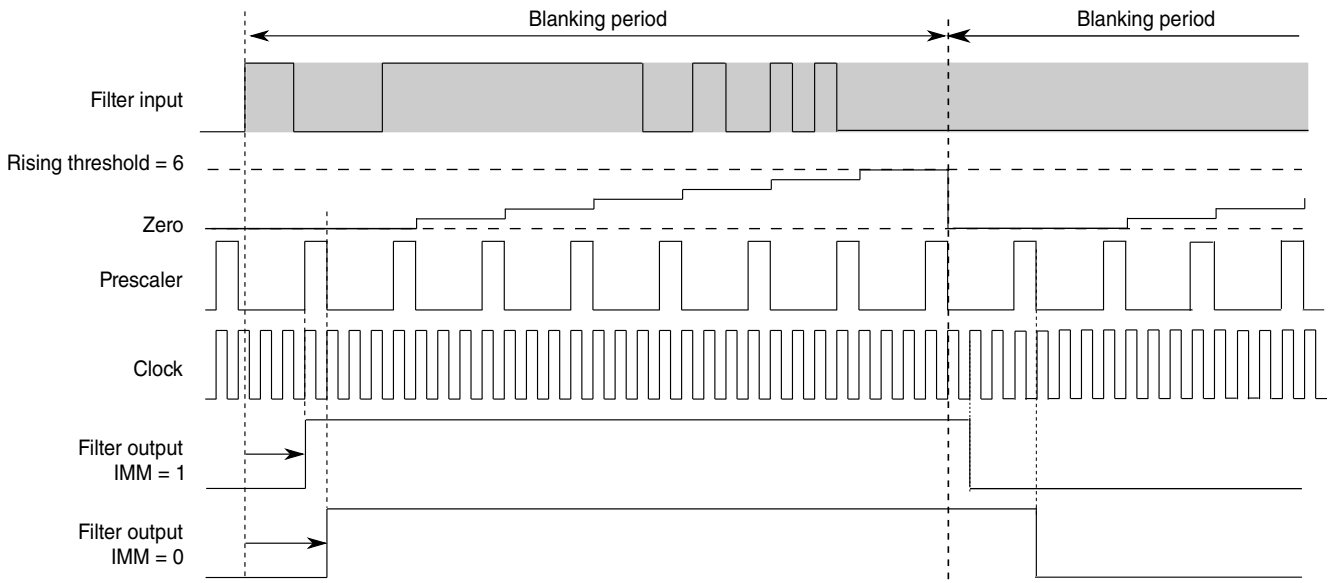


Figure 32-8. Input signal propagation through the filter in windowing filter type

If IMM = 0 then the input signal edges are propagated according to the prescaler ticks. If IMM = 1 then the input edges are propagated immediately (within 3 clock cycles) to the filter output.

32.5.3.6 Application use case 6: prescaler settings in bypass

The prescaler settings influence the timing for an input signal edge to propagate through the filter. In bypass, the IMM bit in the MCR register controls the way an edge is propagated through the filter. If IMM= 0, the prescaled clock is used to sample the input filter signal. The resolution the input signal is sampled is therefore related to the prescaler frequency. Glitches narrower than the prescaler period that are not aligned with the

prescaler clock are not captured by the filter. If IMM = 1, the system clock is used to sample the filter input signal instead of the prescaled clock. In this case, the resolution of the sampled input signal is the system clock.

The figure below describes the filter input signal sampling behavior for IMM = 0 and IMM = 1.

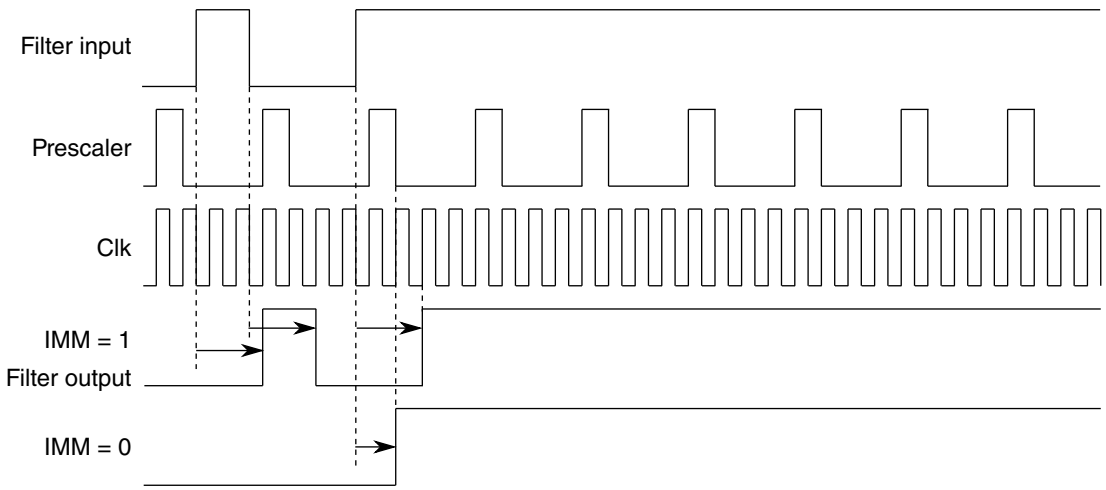


Figure 32-9. Input signal propagation through the filter in bypass

32.5.4 Programming guidelines

The following generalized procedure may be used to configure the Input Glitch Filter Block:

1. Program the SoC pin mux logic to route an external signal to the input of the selected glitch filter.
2. Enable the destination module to receive the Input Glitch filter output.
3. Enable the clock source at the SoC level.
4. Program the Glitch Filter counter prescaler(s) to the desired rate.
5. Select the filter type for each edge using MCR[FFM] and MCR[RFM].
6. Clear any status bits if necessary.
7. Enable the filter using MCR[FGEN]

32.5.5 On-the-fly filter reconfiguration

The architecture of the Input Glitch Filter allows for on the fly reconfiguration of filter parameters such as rise and fall threshold values.

If a parameter is changed it influences the filter operation immediately after the new parameter value was written. The main problem when changing parameters, such as a threshold value, when the filter is enabled is that the filter counter could lose a match due to the parameter change. A great or equal comparator is used in the threshold match logic which allows correct filter operation when the threshold parameters are changed. The figure below shows an example of changing the rising edge parameter value. Note that in this case the match occurs immediately after the parameter is changed since the new parameter value is below the current counter value.

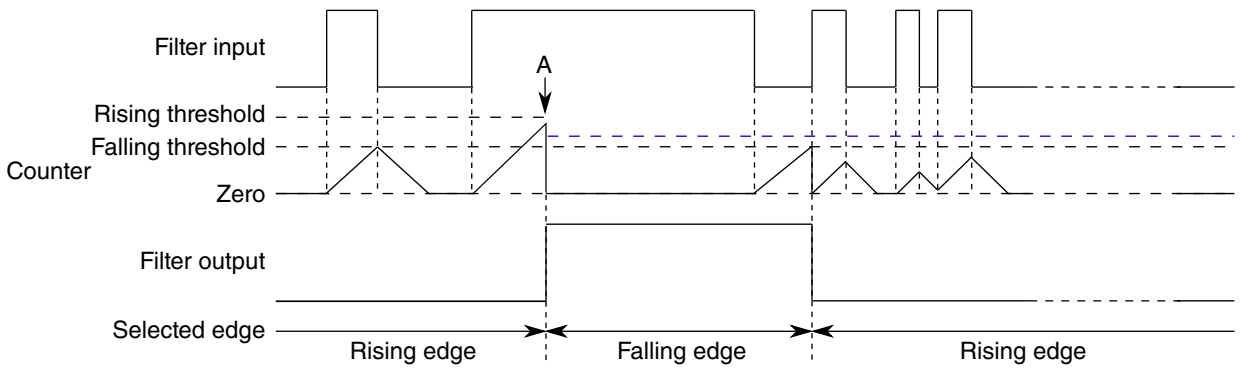


Figure 32-10. On-the-fly rising threshold parameter change

In the case shown in the figure above the match occurs immediately after the new rising threshold is programmed at ‘A’.

NOTE

Note that when changing the threshold values the match may not occur when the counter is equal to the threshold value (the old or the new programmed value). Instead, the match may occur in between the old and new programmed thresholds when the threshold re-programming instant A is done when the counter is greater than the new value. This is a transitory effect which may occur when the threshold value is changed and should be considered by the application.

32.6 Power saving mode

The Input Glitch Filter implements power saving mode thus allowing to reduce power consumption when device inputs connected to the filter are not used or the filter is not necessary to be functioning in a specific SoC mode of operation.

32.6.1 STOP mode

There are two ways for the Input Glitch filter to enter low power mode:

- MDIS bit
- STOP mode system request

The MDIS control bit in the MCR register, when set, disables the filter logic by stopping the module clock signals and thus halting the internal filter counter which reduces the dynamic power consumption of the module. The MDIS bit provides a selective control of the low power operation of each one of the Input Glitch Filter instances in the device.

The STOP mode system request is similar to MDIS operation with exception that it applies to all Input Glitch Filter instances in the device. It does not need to be enabled by any control bit in the Input Glitch filter module. The low power mode in this case is controlled by the assertion of the STOP signal which is a system wide low power control signal.

Chapter 33
HF - Interface

33.1 Chip specific HFIF information

33.1.1 Module Interconnect

This MCU supports **four** HF-interfaces.

The figure below shows an example of how the Input Glitch Filter interacts with on chip peripherals in the HF-Interface use-case. The connecting of the filter output signal to the eMIOS channel is individually connect-able.

The eMIOS channels C28-31 are used for HF-interface usage, because these channels are only from CH type Y (channel type with least supported modes, but high number of available channels on both eMIOS blocks).

To increase the system flexibility several pads are multiplexed to the glitch/symbol filter.

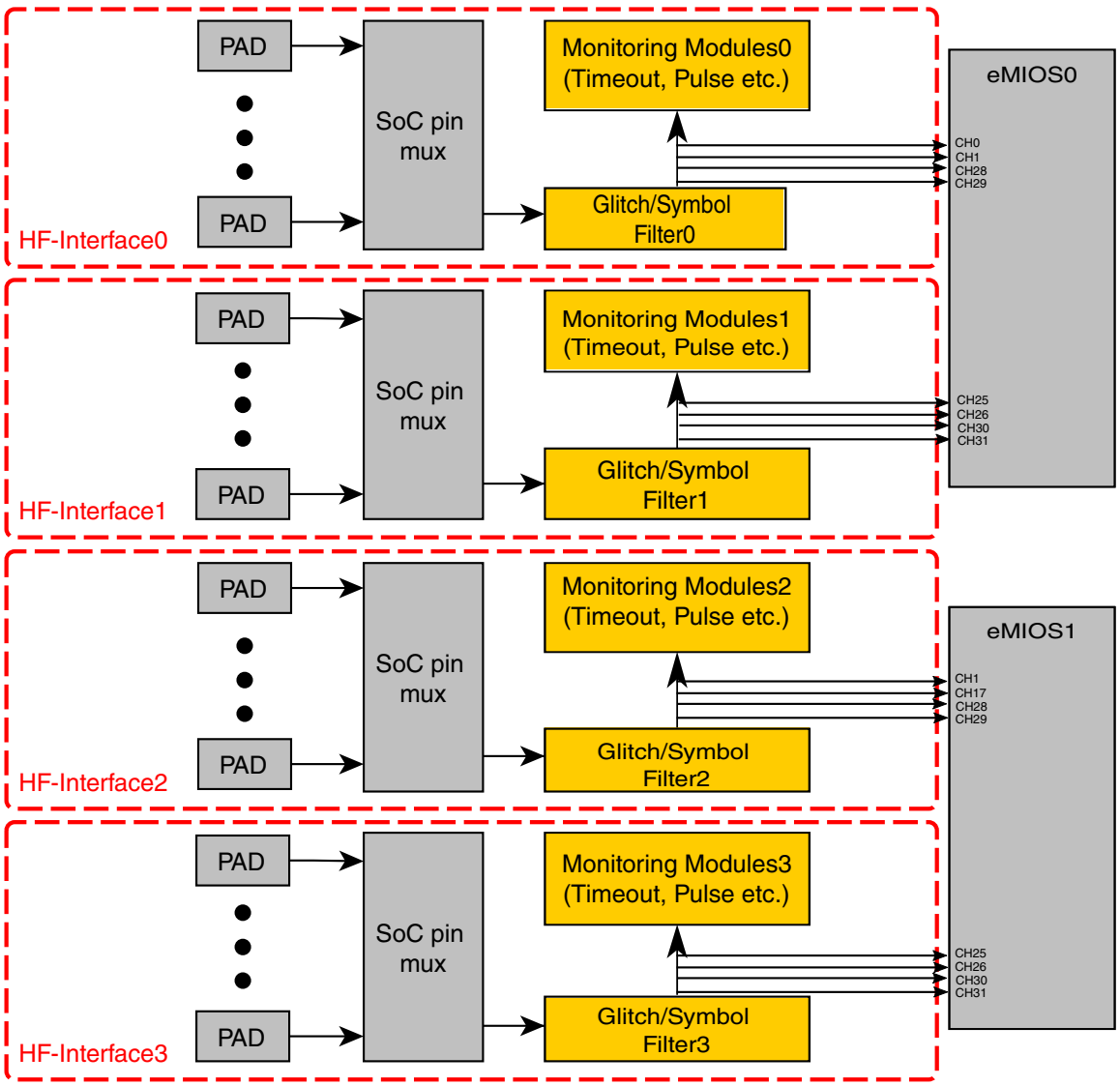


Figure 33-1. Interaction of IGF, HFIF, and eMIOS

The pin muxing implementation for Glitch-Filter0 and eMIOS0 channels 28 and 29 is shown in the diagram below. The input for Glitch-Filter0 is muxed using SIUL-MSCR 1018. The output of the Glicth-Filter0 is directly given as input to the HF-interface0, and is given to eMIOS0 channels 28 and 29 via SIUL MSCRs 540 and 541. When using these channels for HF-interface functionality, the Glitch Filter's outputs must be selected to go to these channels, otherwise normal eMIOS pads can be used. Similar to the muxing below, HF-interface 1, 2 and 3 also have simlar input-muxes using SIUL MSCR registers. This information can be derived using the "IO_Signal_Description_and_Input_multiplexing_tables.xls" attached with the RM.

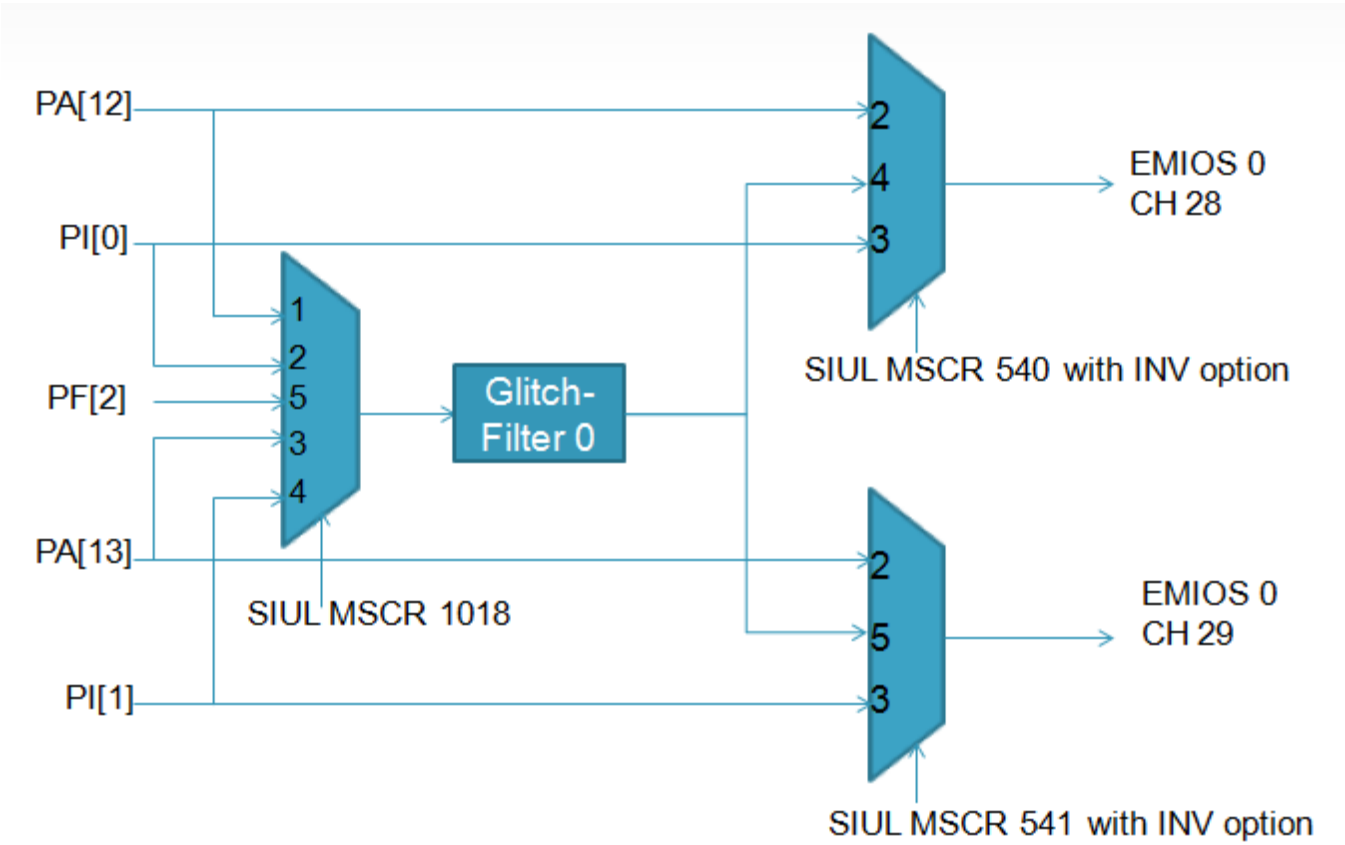


Figure 33-2. Pin muxing implementation for Glitch-Filter0 and eMIOS0 channel 28 and 29

33.2 Introduction

The HF-Interface is used to establish a cost-effective internal vehicle communication between MCU(s) and antenna modules. The HF interface is a single wire bus and offers a bi-directional communication with a maximum bandwidth of 20kBaud. Antenna modules could be: Keyless-go, Remote control, Key fob, Tire pressure. The antenna modules are responsible for the high-frequency wireless communication and HFIF offers a custom specific communication interface inside the vehicle.

The goal is to support the HF-interface in this MCU and reduce the load on Core significantly. Active communication happens before and while the vehicle is moving.

33.3 Block Diagram

The figure below is the block diagram for the HFIF module.

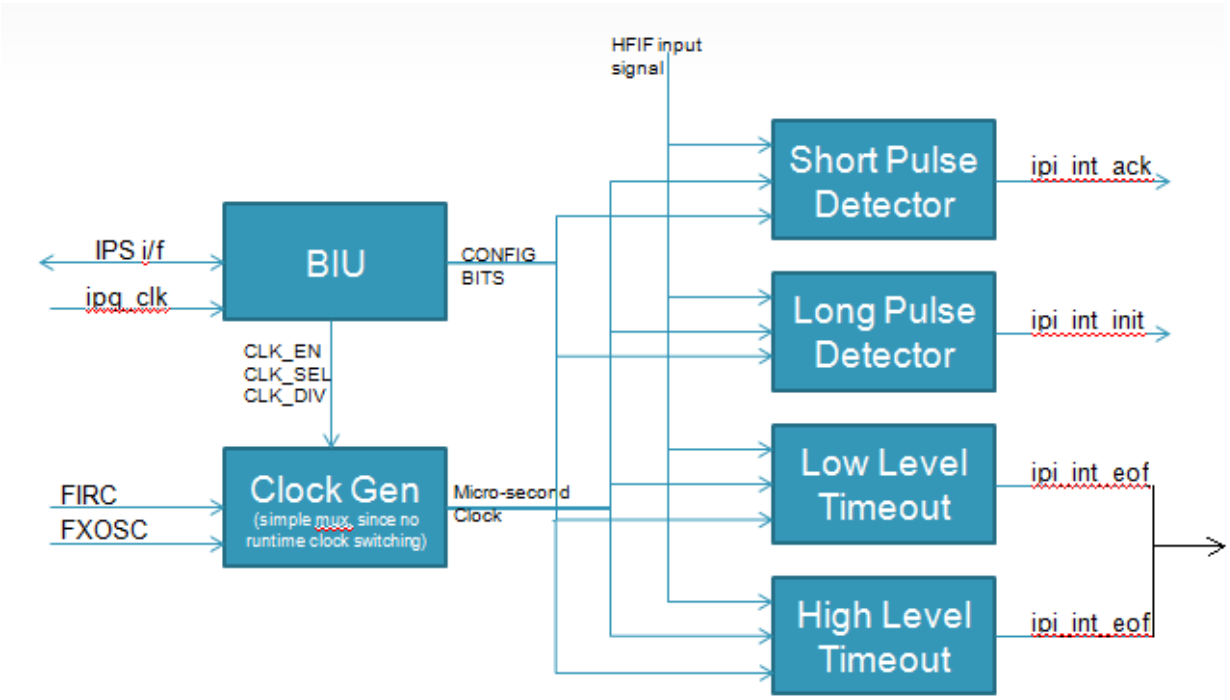


Figure 33-3. HFIF Block Diagram.

The following modules form the part of HFIF IP.

- Long Pulse Detection.
- Short Pulse Detection.
- Low Level Timeout.
- High Level Timeout.
- Clock Generation.

These modules are used in combination with the following IPs to provide the complete HF-interface support:

- Input Glitch Filter.
- eMIOS (for Rx-edge sampling and Tx handling).
- DMA channels/task to transfer the data to and fro.
- INTC (for interrupt handling).

The following diagram shows how the HF-Interface could be supported by these resources when used along with the HFIF timer and pulse-detector modules.

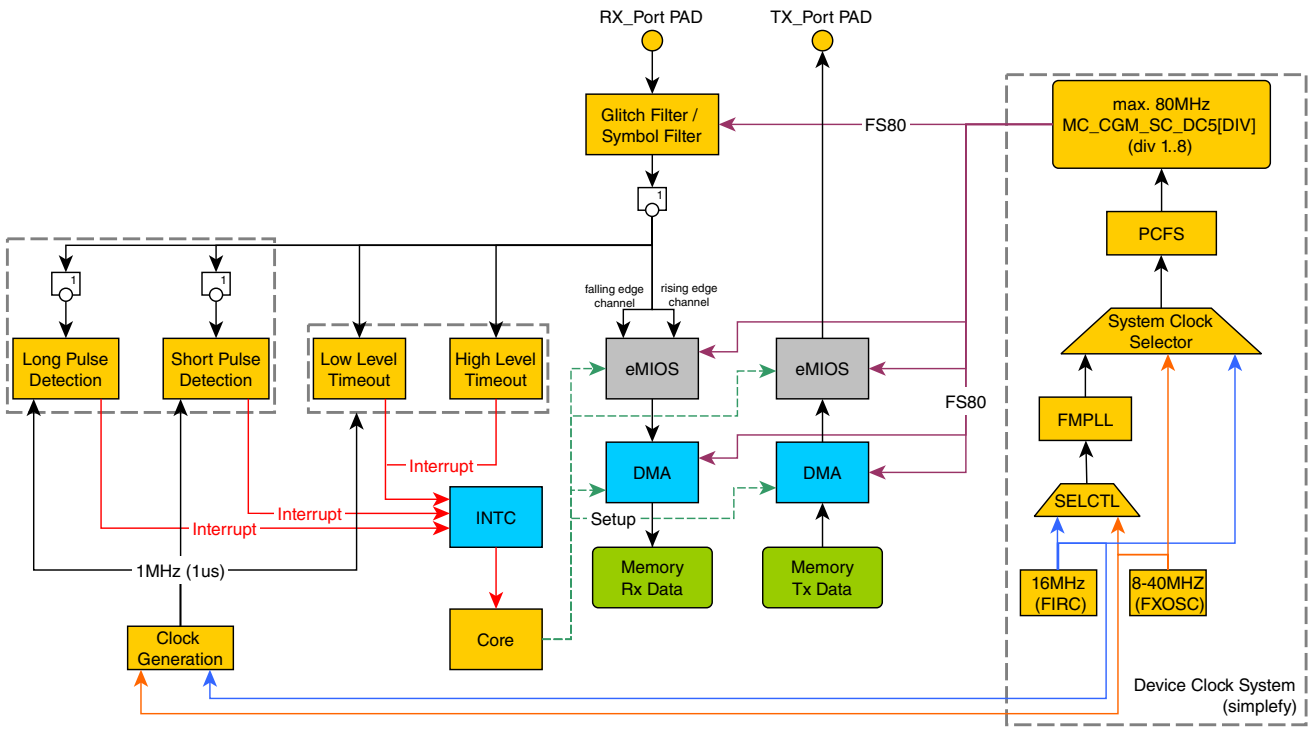


Figure 33-4. Modules used in the MCU to support HF-interface.

The next section describes the various tasks performed by these modules to acheive the functionality of HF-interface.

33.4 Overview

33.4.1 Receiving Data

The following tasks are needed to perform the Rx Function of HF-interface.

Init Tasks:

- Setup glitch/symbol filter
- Setup timeout modules
- Setup pulse detection modules
- Setup DMA to transfer the capture values from the eMIOS register to memory
- Setup 2x eMIOS channels in the "single action input capture" (SAIC) mode, one for the falling and the other one for the rising edge

Main Tasks:

- Glitch/symbol filter output signal is routed to two eMIOS channels, one channel will capture the time of rising and the other the falling edge. Together with the timing the signal polarity will be captured and published in the register interface.
- The DMA task is triggered when one of the eMIOS channels capture a new time. The DMA moves the timing values to the memory. The DMA can trigger an interrupt to the core when the maximum number of transfers is reached.
- The monitor modules (timeout and pulse detection) could be enabled/disabled (depending of the protocol status) by the core and trigger an interrupt in case one of the monitoring events occurs.

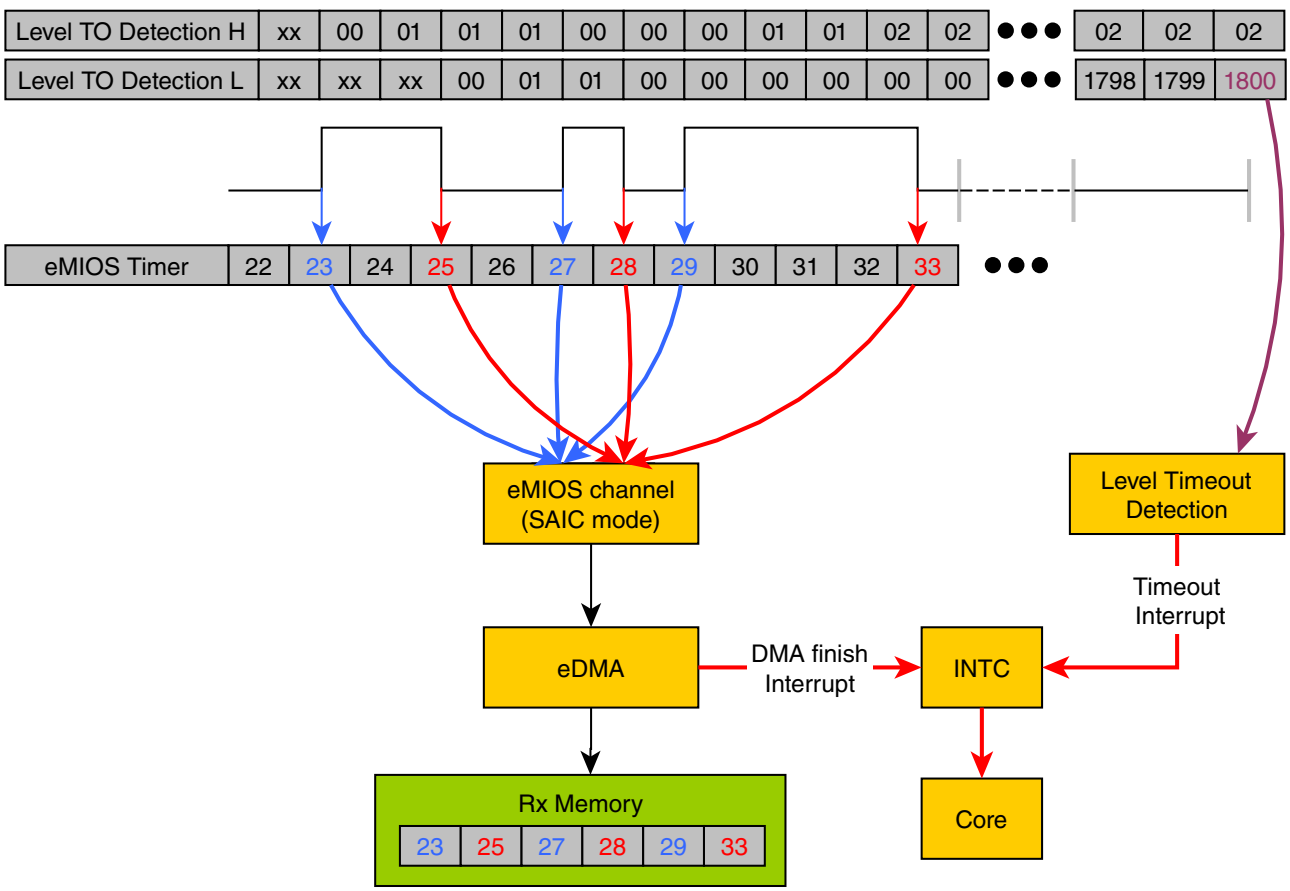


Figure 33-5. Data Flow for HFIF Rx function.

The Glitch Filter filters out any signal noise on its input port during a specific time and publishes the filtered signal to its output port. Various filtering schemes are supported by the Glitch Filter. One of them is "Windowing filter with post sample". In case the input signal pass the decision level (high or low) the timer will start to count. During this filtering time, the output port will still be at the previous signal level. If the filter time expires the output port will follow the actual input port level. In case the signal level goes

back to the previous one before the filter time expires, no signal level change is visible on the output port. The following waveform shows the behavior of the glitch filter in "Windowing filter with post sample" mode.

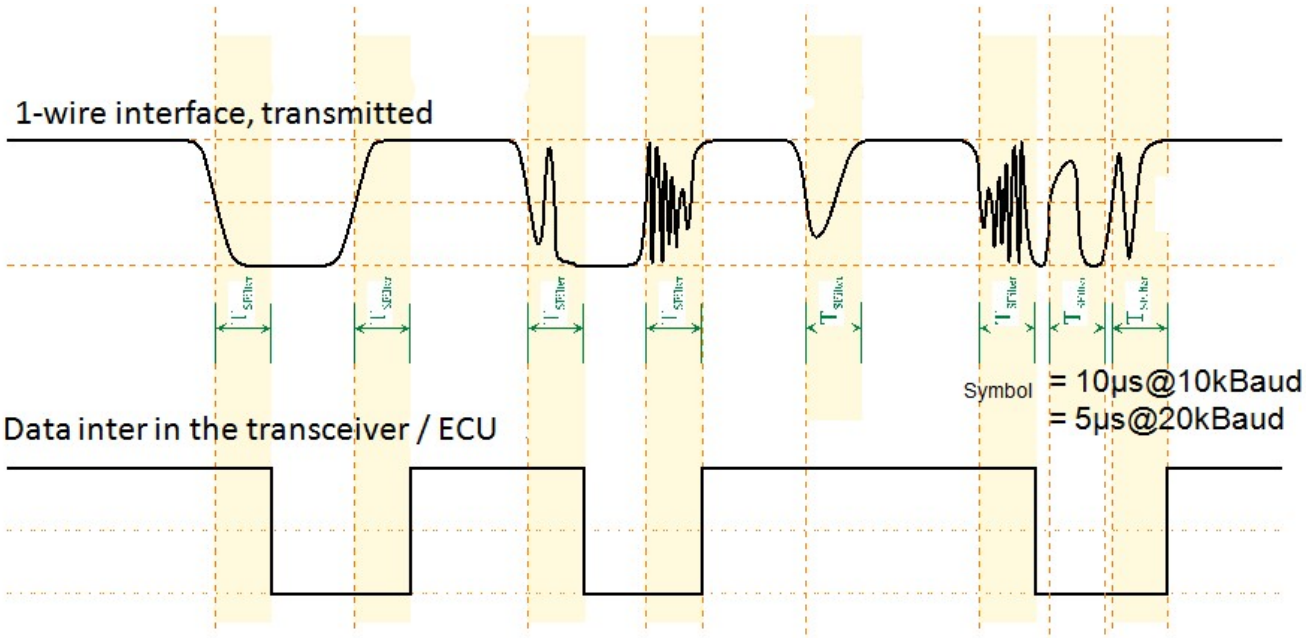


Figure 33-6. Windowing Filter with Post Sample.

33.4.2 Transmitting Data

The following tasks are needed to perform the Tx Function of HF-interface.

Init Tasks:

- Setup eMIOS for Single Action Output Compare (SAOC) mode, setup signal polarity (EDPOL register)
- Setup DMA to copy Tx-data from memory into the eMIOS register A1

Main Tasks:

- Core: WritesTx message data into the memory
- Core: Write the first Tx message data to eMIOS registers
- eMIOS: Triggers the DMA task to copy the next values to eMIOS registers
- DMA: send " end-of-transfer" via interrupt to the core when the complete data set (Tx message) is copied

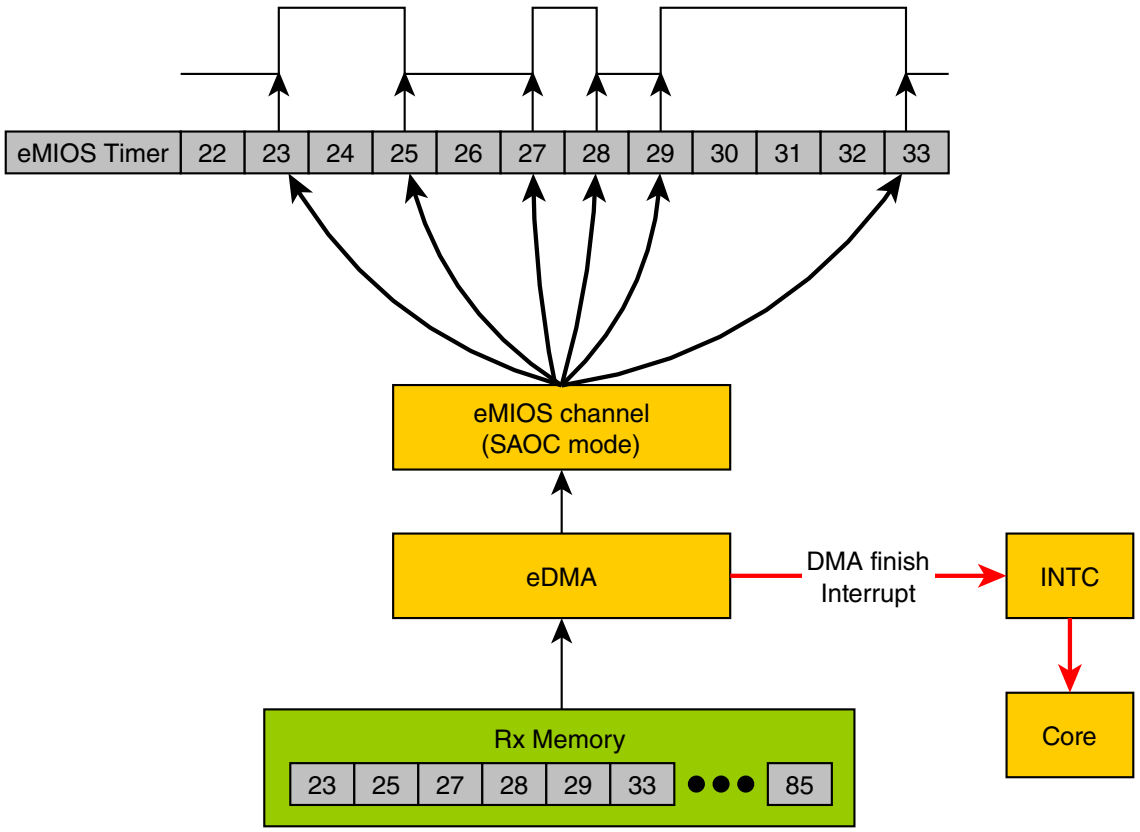


Figure 33-7. Data Flow for HFIF Tx function.

33.5 Functional Description

33.5.1 Clock Generation

The clock generation module is used to generate an clock for the pulse and timeout sub-modules of the HF interface. The customer can choose between the internal FIRC (16MHz) or the external FXOSC (8-40MHz) as clock source.

The clock generation module has a clock pre-scaler to reduce the input clock to a lower level (e.g 1MHz/1usec cycle time). To control the module function, the Clock Configuration Register (CCR) is used.

33.5.2 Signal Level Timeout Detection

The timeout detection is based on two timer modules one for the low-level and one for the high-level detection. The counter counts as long as the level is stable on the signal input. A falling edge resets the timer to measure the low level and a rising edge resets the timer for the high signal level. If one of both timer reach the maximum/timeout value the module triggers an interrupt. Both modules share the same interrupt line.

To control these modules, the following configuration registers are used :

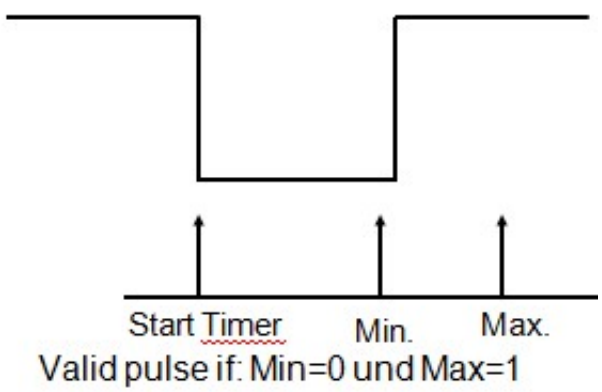
- Low-level Timeout Detector Configuration Register (LTDCR)
- High-level Timeout Detector Configuration Register (HTDCR)

33.5.3 Pulse Detection

The HF-Interface protocol defines two signal pulses of a specific length. A short ACK pulse and a long INIT pulse (sent two times in a sequence).

The pulse are only valid in case the length fits in the specified timing window. For this the customer can specify a minimum and a maximum value as the following graphic shows. When the signal fits the timing window (min-max-window) an interrupt will be triggered to the core.

In case the pulse length is outside of the min-max-window (before and after) the pulse is ignored and no interrupt event is triggered.



Due to the fact that two different pulse length must be supported, two instances of the same module are used. The typical working timing ranges for the ACK (typical range of 90-110 usec) and the INIT (typical range of 3.9 - 4.1 msec) pulse are different.

NOTE

The modules are not able to detect a sequence of two or more pulses (e.g. INIT), for this each single INIT pulse will trigger an interrupt.

To control these modules, the following configuration registers are used :

- Short Pulse Detector Configuration Register (SPDCR)
- Long Pulse Detector Configuration Register (LPDCR)

33.5.4 Interrupts

Timeout (1x) and Pulse (2x) detection modules offers three new interrupt signals. They are managed via these two registers.

- Interrupt Enable Register (IER)
- Interrupt Status Register (ISR)

By disabling the interrupt will have similar effect like to disable the modules.

33.6 Memory Map and Registers

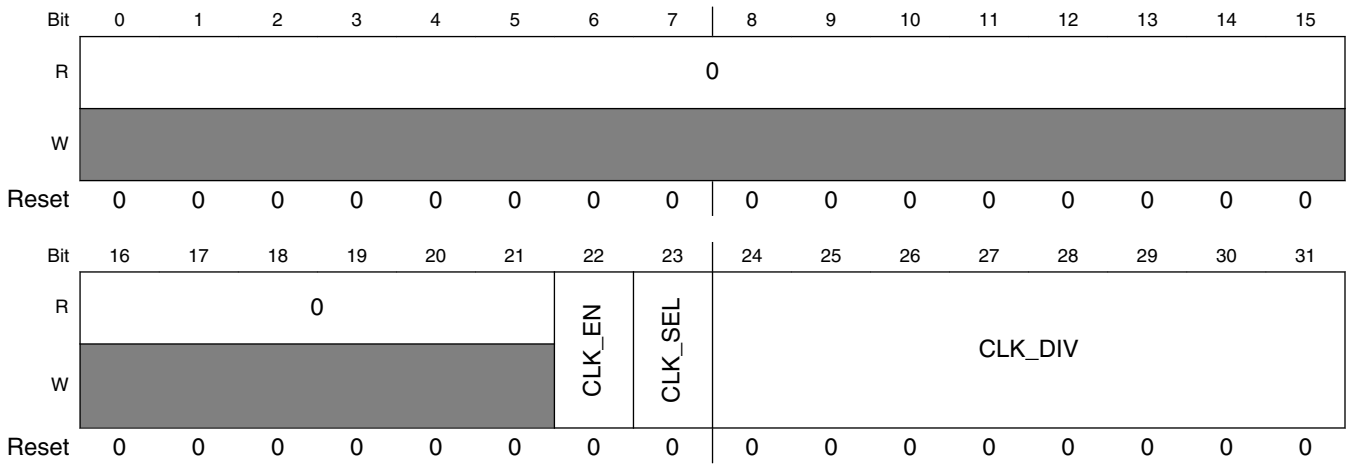
HFInterface memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Clock Configuration Register (HFInterface_CCR)	32	R/W	0000_0000h	33.6.1/711
4	Short Pulse Detector Configuration Register (HFInterface_SPDCR)	32	R/W	0000_0000h	33.6.2/712
8	Long Pulse Detector Configuration Register (HFInterface_LPDCR)	32	R/W	0000_0000h	33.6.3/713
C	Low-level Timeout Detector Configuration Register (HFInterface_LTDCR)	32	R/W	0000_0000h	33.6.4/714
10	High-level Timeout Detector Configuration Register (HFInterface_HTDCR)	32	R/W	0000_0000h	33.6.5/715
14	Interrupt Enable Register (HFInterface_IER)	32	R/W	0000_0000h	33.6.6/716
18	Interrupt Status Register (HFInterface_ISR)	32	R/W	0000_0000h	33.6.7/717

33.6.1 Clock Configuration Register (HFInterface_CCR)

This register is used for generating clock for the pulse and time out sub modules of the HF interface. Internal FIRC (16MHz) or the external FXOSC (8-40MHz) can be used as clock source.

Address: 0h base + 0h offset = 0h



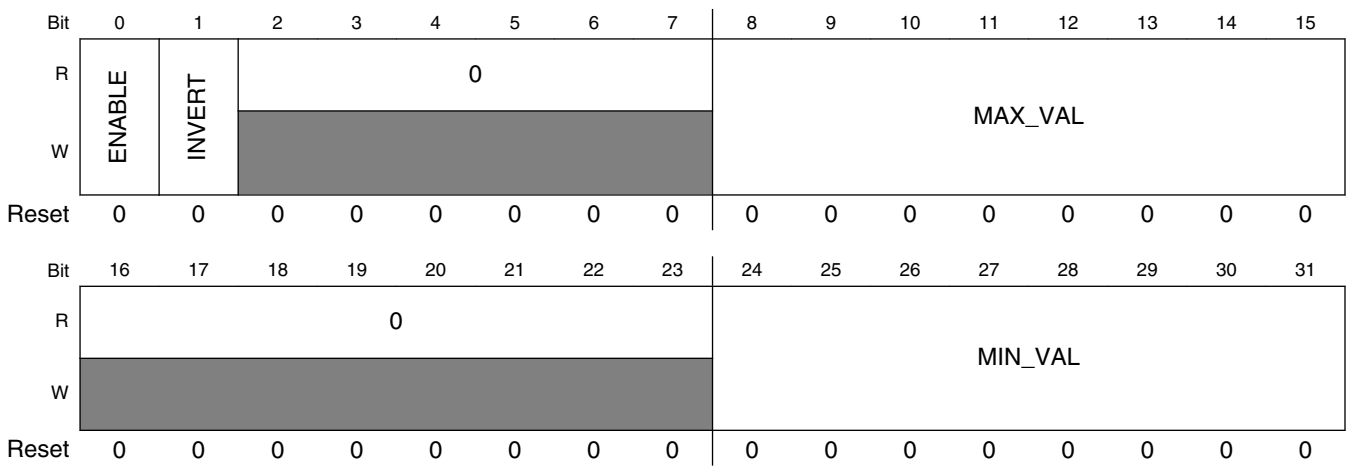
HFInterface_CCR field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 CLK_EN	Clock Enable This bit controls the output us_clk which will be enabled only after this bit is made 1.
23 CLK_SEL	Clock Select This bit selects the clock source. 0 FXOSC 1 FIRC
24–31 CLK_DIV	Clock Divider An M MHz clock will be needed to be divided by M to get 1us clock.

33.6.2 Short Pulse Detector Configuration Register (HFInterface_SPDCR)

This register defines a short ACK pulse. The pulse are only valid in case the length fits in the specified timing window. When the signal fits the timing window (min-max-window) an interrupt will be triggered to the core. In case the pulse length in outside of the minmax- window (before and after) the pulse is ignored an no interrupt event is triggered.

Address: 0h base + 4h offset = 4h



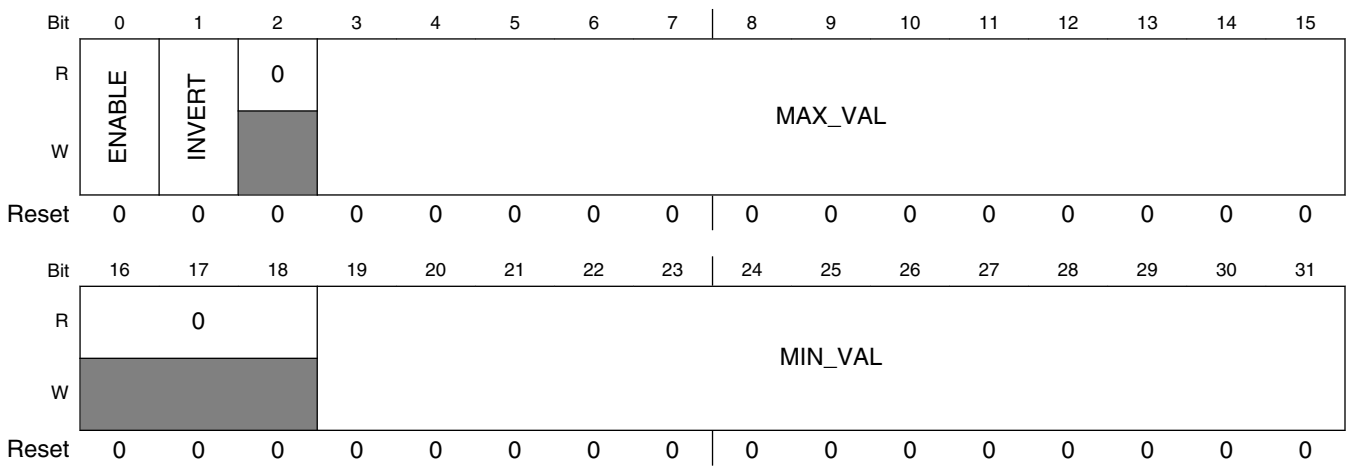
HFInterface_SPDCR field descriptions

Field	Description
0 ENABLE	This bit enables the input signal.
1 INVERT	This bit inverts the input signal
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 MAX_VAL	Maximum clock cycles Specify the maximum number of clock cycles after the the edge for a valid pulse.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 MIN_VAL	Minimum clock cycles Specify the minimum number of clock cycles after the the edge for a valid pulse.

33.6.3 Long Pulse Detector Configuration Register (HFInterface_LPDCR)

This register defines a long INIT pulse. The pulse are only valid in case the length fits in the specified timing window. When the signal fits the timing window (min-max-window) an interrupt will be triggered to the core. In case the pulse length in outside of the minmax- window (before and after) the pulse is ignored an no interrupt event is triggered.

Address: 0h base + 8h offset = 8h



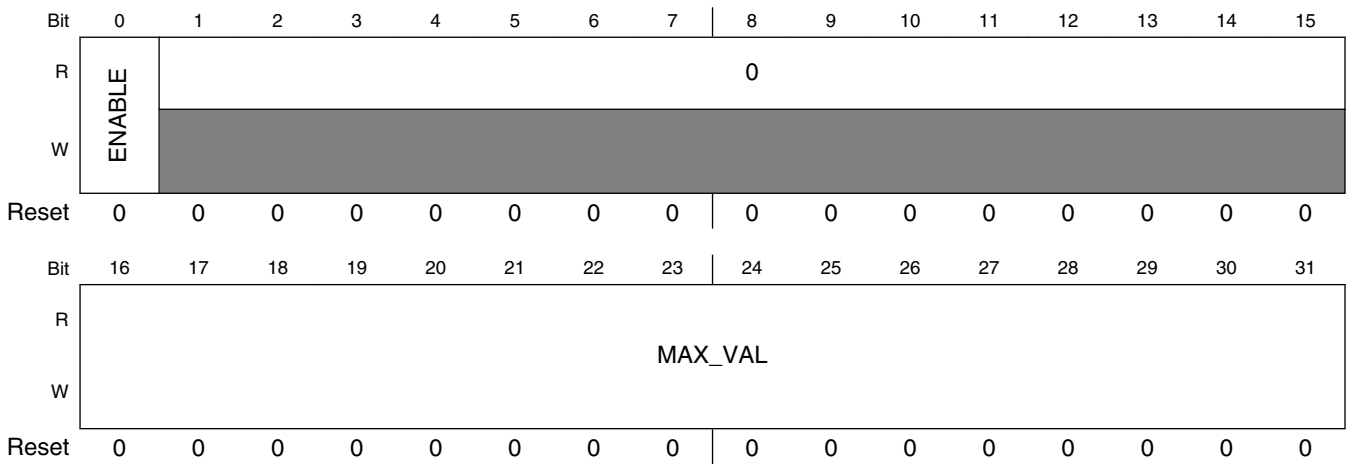
HFInterface_LPDCR field descriptions

Field	Description
0 ENABLE	This bit enables the input signal.
1 INVERT	This bit inverts the input signal
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–15 MAX_VAL	Maximum high level value Gives the maximum number of clock cycles for a high level time out.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–31 MIN_VAL	Minimum high level value Gives the minimum number of clock cycles for a high level time out.

33.6.4 Low-level Timeout Detector Configuration Register (HFInterface_LTDCR)

This register controls the low level time out detection. The counter counts as long as the level is stable on the signal input and a falling edge resets the timer to measure the low level time out.

Address: 0h base + Ch offset = Ch



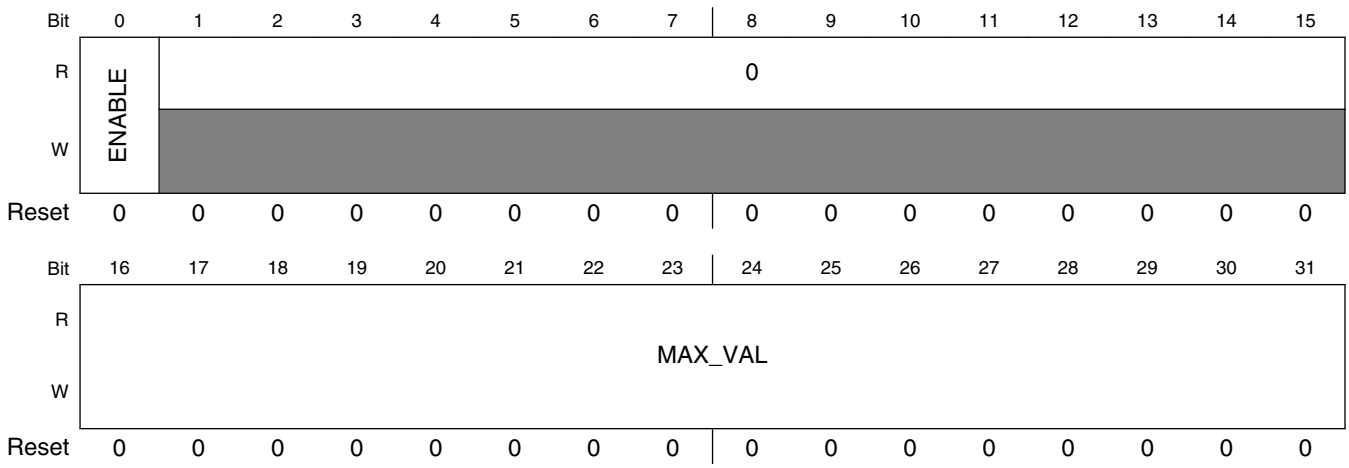
HFInterface_LTDCR field descriptions

Field	Description
0 ENABLE	This bit enables the input signal.
1–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 MAX_VAL	Maximum value Gives the maximum number of clock cycles for a low level time out.

33.6.5 High-level Timeout Detector Configuration Register (HFInterface_HTDCR)

This register controls the high level time out detection. The counter counts as long as the level is stable on the signal input and a rising edge resets the timer for the high signal level.

Address: 0h base + 10h offset = 10h



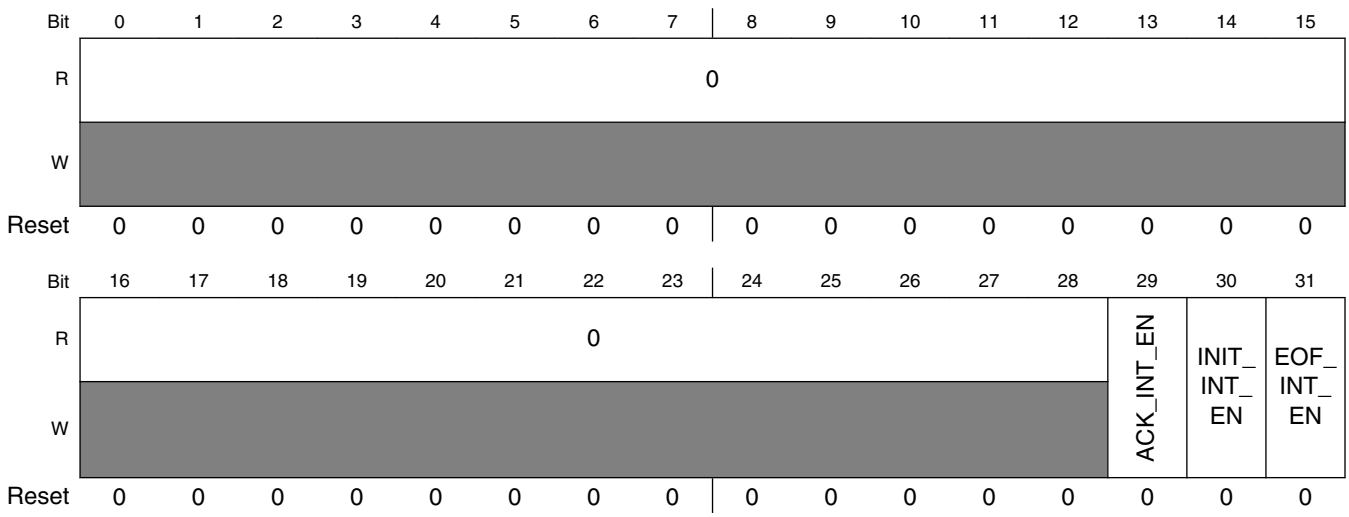
HFInterface_HTDCR field descriptions

Field	Description
0 ENABLE	This bit enables the input signal.
1–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 MAX_VAL	Maximum value Gives the maximum number of clock cycles for a high level time out.

33.6.6 Interrupt Enable Register (HFInterface_IER)

This register enables the interrupt signal.

Address: 0h base + 14h offset = 14h



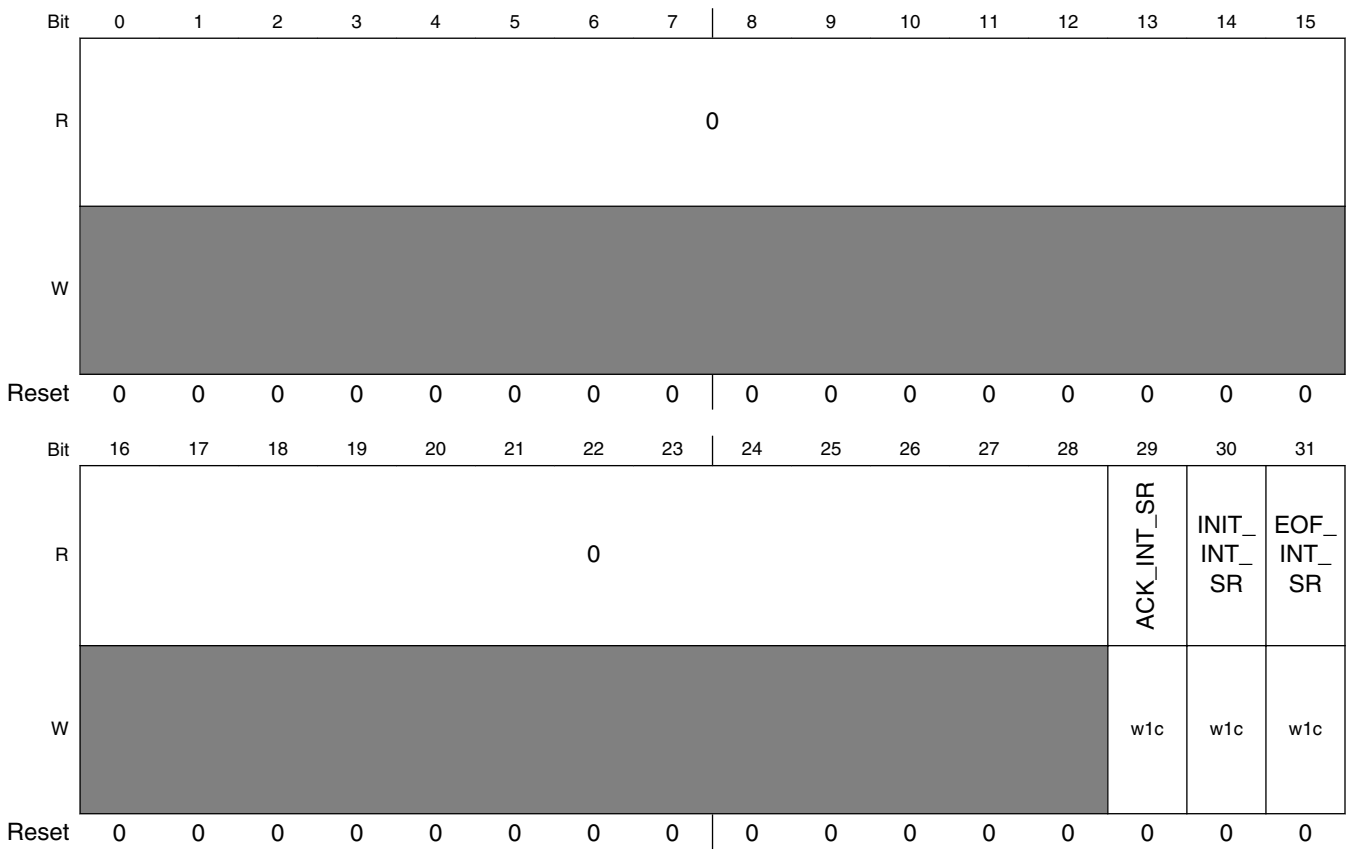
HFInterface_IER field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 ACK_INT_EN	Interrupt enable for ACK pulse interrupt.
30 INIT_INT_EN	Interrupt enable for INIT pulse interrupt.
31 EOF_INT_EN	Interrupt enable for level-timeout interrupt (low or high).

33.6.7 Interrupt Status Register (HFInterface_ISR)

This register gives the status of interrupt signals.

Address: 0h base + 18h offset = 18h



HFInterface_ISR field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 ACK_INT_SR	Interrupt status for ACK pulse interrupt.
30 INIT_INT_SR	Interrupt status for INIT pulse interrupt.
31 EOF_INT_SR	Interrupt status for level-timeout interrupt (low or high).

Chapter 34
Analog-to-Digital Converter (ADC)

34.1 Chip-specific ADC information

34.1.1 ADC configuration

The MPC5746C supports two independent SAR ADCs: one 10-bit ADC (ADC0) and one 12-bit ADC (ADC1).

NOTE

- ADC0 and ADC1 have differences in their features and control registers. See the following sections for details.
- See the *MPC5746C Microcontroller Data Sheet* for performance specifications.

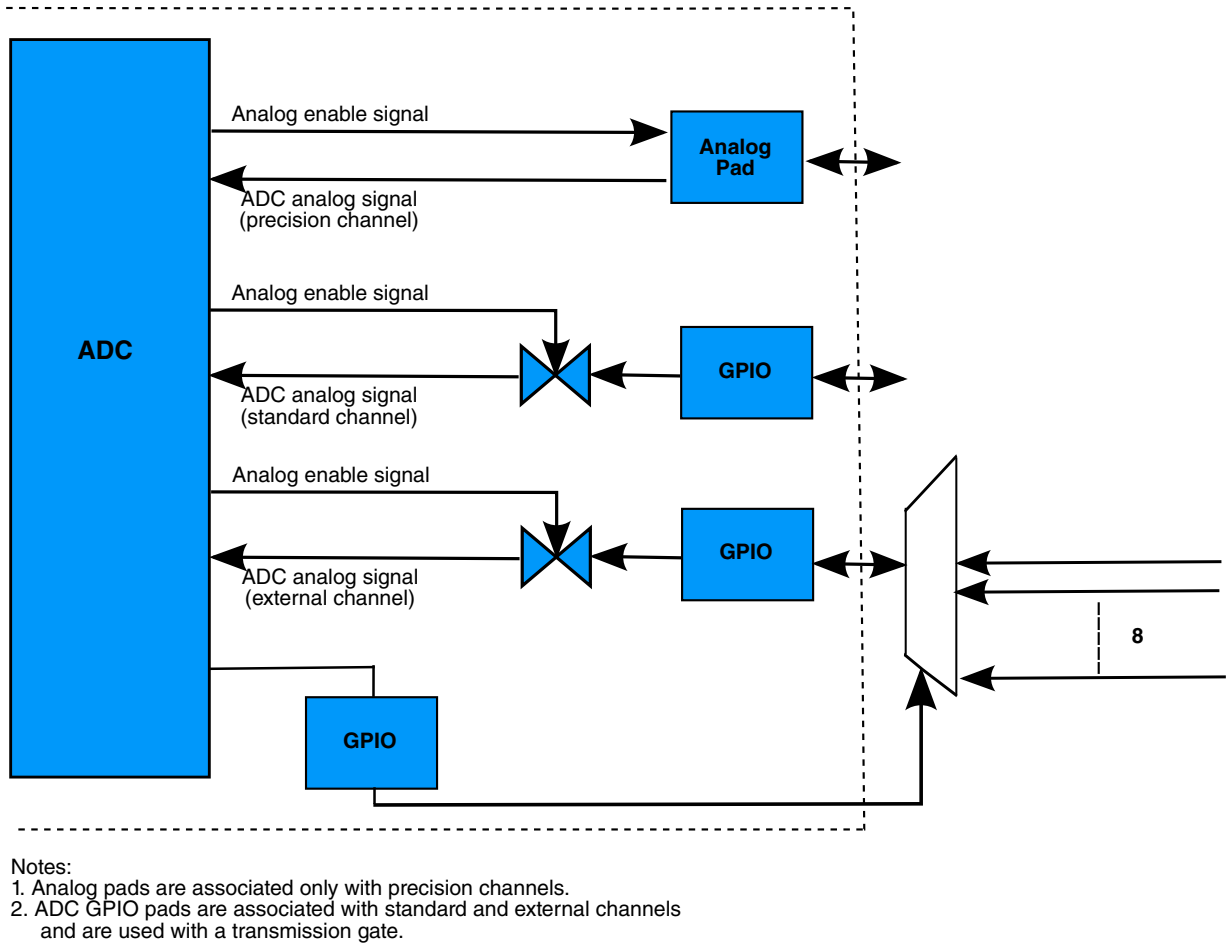
34.1.2 Feature comparison

The following table lists the features for each of the on-chip SAR ADC modules. Refer Figure 34-1 for ADC channel mapping details with various pads.

Table 34-1. ADC features

Feature	ADC_0 (10-bit)	ADC_1 (12-bit)
Resolution	10-bit	12-bit
No. of precision channels	0	16
No. of standard internal channels	36	16
No. of special internal channels	5	5
No. of external channels	32	32
BCTU trigger support	yes	yes
DMA support	yes	yes
Presampling support	yes	yes
No. of watchogs	6	3

Figure 34-1 shows the difference between standard, precision and external channels.



Notes:
1. Analog pads are associated only with precision channels.
2. ADC GPIO pads are associated with standard and external channels and are used with a transmission gate.

Figure 34-1. ADC channel types

34.1.3 ADC register reset values

Reset values for the following ADC registers varies per the ADC instance. This value overrides the reset value that is shown in the body of this chapter.

Table 34-2. ADC register reset values[1]

Register	ADC_0 Reset Value (10-bit ADC)	ADC_1 Reset Value (12-bit ADC)
Threshold Register 0 (THRHLR0)	03FF_0000h	0FFF_0000h
Threshold Register 1 (THRHLR1)	03FF_0000h	0FFF_0000h
Threshold Register 2 (THRHLR2)	03FF_0000h	0FFF_0000h
Threshold Register 3 (THRHLR3)	03FF_0000h	—[1]
Threshold Register 4 (THRHLR4)	03FF_0000h	—[1]

Table continues on the next page...

Table 34-2. ADC register reset values[1] (continued)

Register	ADC_0 Reset Value (10-bit ADC)	ADC_1 Reset Value (12-bit ADC)
Threshold Register 5 (THRHLR5)	03FF_0000h	—[1]
Calibration, BIST Control and status register (CALBISTREG)	4137_06F6h	C73706F6h
Conversion Timing Register 0 (CTR0)	0000_0014h	0000_0014h
Conversion Timing Register 1 (CTR1)	0000_0014h	0000_0014h
Conversion Timing Register 2 (CTR2)	0000_0014h	0000_0014h

1. ADC_1 does not include the THRHLR3,THRHLR4, and THRHLR5 registers.

34.1.4 SARADC control registers

Not all control registers are applicable to every ADC instance. The following table shows the registers applicable to each.

Table 34-3. SAR ADC control register applicability

Register	ADC_0	ADC_1
Main Configuration Register (ADC_MCR)	yes	yes
Main Status register (ADC_MSR)	yes	yes
Interrupt Status Register (ADC_ISR)	yes	yes
Channel Pending register 0 (ADC_CEOCFR0)	yes	yes
Channel Pending register 1 (ADC_CEOCFR1)	yes	yes
Channel Pending register 2 (ADC_CEOCFR2)	yes	yes
Interrupt Mask Register (ADC_IMR)	yes	yes
Channel Interrup Mask Register 0 (ADC_CIMR0)	yes	yes
Channel Interrup Mask Register 1 (ADC_CIMR1)	yes	yes
Channel Interrup Mask Register 2 (ADC_CIMR2)	yes	yes
Watchdog Threshold Interrupt Status Register (ADC_WTISR)	yes	yes
Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)	yes	yes
DMA Enable register (ADC_DMAE)	yes	yes
DMA Channel Select Register 0 (ADC_DMAR0)	yes	yes
DMA Channel Select Register 1 (ADC_DMAR1)	yes	yes
DMA Channel Select Register 2 (ADC_DMAR2)	yes	yes
Threshold Register (ADC_THRHLR0)	yes	yes
Threshold Register (ADC_THRHLR1)	yes	yes
Threshold Register (ADC_THRHLR2)	yes	yes
Threshold Register (ADC_THRHLR3)	yes	no
Presampling Control Register (ADC_PSCR)	yes	yes
Presampling register 0 (ADC_PSR0)	yes	yes
Presampling register 1 (ADC_PSR1)	yes	yes

Table continues on the next page...

Table 34-3. SAR ADC control register applicability (continued)

Register	ADC_0	ADC_1
Presampling register 2 (ADC_PSR2)	yes	yes
Conversion Timing Register 0 (ADC_CTR0)	yes	yes
Conversion Timing Register 1 (ADC_CTR1)	yes	yes
Conversion Timing Register 2 (ADC_CTR2)	yes	yes
Channel Pending register 0 (ADC_NCMR0)	yes	yes
Channel Pending register 1 (ADC_NCMR1)	yes	yes
Channel Pending register 2 (ADC_NCMR2)	yes	yes
Injected Conversion Mask Register 0 (ADC_JCMR0)	yes	yes
Injected Conversion Mask Register 1 (ADC_JCMR1)	yes	yes
Injected Conversion Mask Register 2 (ADC_JCMR2)	yes	yes
Decode Signals Delay Register (ADC_DSDR)	yes	yes
Power Down Exit Delay Register (ADC_PDEDR)	Yes	Yes
Channel Data Register (ADC_CDR0)	no	yes
Channel Data Register (ADC_CDR1)	no	yes
Channel Data Register (ADC_CDR2)	no	yes
Channel Data Register (ADC_CDR3)	yes	yes
Channel Data Register (ADC_CDR4)	yes	yes
Channel Data Register (ADC_CDR5)	no	yes
Channel Data Register (ADC_CDR6)	no	yes
Channel Data Register (ADC_CDR7)	yes	yes
Channel Data Register (ADC_CDR8)	yes	yes
Channel Data Register (ADC_CDR9)	no	yes
Channel Data Register (ADC_CDR10)	no	yes
Channel Data Register (ADC_CDR11)	yes	yes
Channel Data Register (ADC_CDR12)	yes	yes
Channel Data Register (ADC_CDR13)	no	yes
Channel Data Register (ADC_CDR14)	no	yes
Channel Data Register (ADC_CDR15)	no	yes
Channel Data Register (ADC_CDR16)	no	no
Channel Data Register (ADC_CDR17)	no	no
Channel Data Register (ADC_CDR18)	yes	no
Channel Data Register (ADC_CDR19)	yes	no
Channel Data Register (ADC_CDR20)	yes	no
Channel Data Register (ADC_CDR21)	yes	no
Channel Data Register (ADC_CDR22)	no	no
Channel Data Register (ADC_CDR23)	no	no
Channel Data Register (ADC_CDR24)	no	no
Channel Data Register (ADC_CDR25)	no	no
Channel Data Register (ADC_CDR26)	no	no

Table continues on the next page...

Table 34-3. SAR ADC control register applicability (continued)

Register	ADC_0	ADC_1
Channel Data Register (ADC_CDR27)	no	no
Channel Data Register (ADC_CDR28)	no	no
Channel Data Register (ADC_CDR29)	no	no
Channel Data Register (ADC_CDR30)	no	no
Channel Data Register (ADC_CDR31)	no	no
Channel Data Register (ADC_CDR32)	yes	yes
Channel Data Register (ADC_CDR33)	yes	yes
Channel Data Register (ADC_CDR34)	yes	yes
Channel Data Register (ADC_CDR35)	yes	yes
Channel Data Register (ADC_CDR36)	no	yes
Channel Data Register (ADC_CDR37)	no	yes
Channel Data Register (ADC_CDR38)	yes	yes
Channel Data Register (ADC_CDR39)	yes	yes
Channel Data Register (ADC_CDR40)	yes	yes
Channel Data Register (ADC_CDR41)	yes	yes
Channel Data Register (ADC_CDR42)	yes	yes
Channel Data Register (ADC_CDR43)	yes	yes
Channel Data Register (ADC_CDR44)	yes	yes
Channel Data Register (ADC_CDR45)	yes	yes
Channel Data Register (ADC_CDR46)	yes	yes
Channel Data Register (ADC_CDR47)	yes	yes
Channel Data Register (ADC_CDR48)	yes	no
Channel Data Register (ADC_CDR49)	yes	no
Channel Data Register (ADC_CDR50)	yes	yes
Channel Data Register (ADC_CDR51)	yes	yes
Channel Data Register (ADC_CDR52)	yes	yes
Channel Data Register (ADC_CDR53)	yes	yes
Channel Data Register (ADC_CDR54)	yes	no
Channel Data Register (ADC_CDR55)	yes	no
Channel Data Register (ADC_CDR56)	yes	no
Channel Data Register (ADC_CDR57)	yes	no
Channel Data Register (ADC_CDR58)	yes	no
Channel Data Register (ADC_CDR59)	yes	no
Channel Data Register (ADC_CDR60)	yes	no
Channel Data Register (ADC_CDR61)	yes	no
Channel Data Register (ADC_CDR62)	yes	no
Channel Data Register (ADC_CDR63)	yes	no
Channel Data Register (External Channels) (ADC_CDR64)	yes	yes
Channel Data Register (External Channels) (ADC_CDR65)	yes	yes

Table continues on the next page...

Table 34-3. SAR ADC control register applicability (continued)

Register	ADC_0	ADC_1
Channel Data Register (External Channels) (ADC_CDR66)	yes	yes
Channel Data Register (External Channels) (ADC_CDR67)	yes	yes
Channel Data Register (External Channels) (ADC_CDR68)	yes	yes
Channel Data Register (External Channels) (ADC_CDR69)	yes	yes
Channel Data Register (External Channels) (ADC_CDR70)	yes	yes
Channel Data Register (External Channels) (ADC_CDR71)	yes	yes
Channel Data Register (External Channels) (ADC_CDR72)	yes	yes
Channel Data Register (External Channels) (ADC_CDR73)	yes	yes
Channel Data Register (External Channels) (ADC_CDR74)	yes	yes
Channel Data Register (External Channels) (ADC_CDR75)	yes	yes
Channel Data Register (External Channels) (ADC_CDR76)	yes	yes
Channel Data Register (External Channels) (ADC_CDR77)	yes	yes
Channel Data Register (External Channels) (ADC_CDR78)	yes	yes
Channel Data Register (External Channels) (ADC_CDR79)	yes	yes
Channel Data Register (External Channels) (ADC_CDR80)	yes	yes
Channel Data Register (External Channels) (ADC_CDR81)	yes	yes
Channel Data Register (External Channels) (ADC_CDR82)	yes	yes
Channel Data Register (External Channels) (ADC_CDR83)	yes	yes
Channel Data Register (External Channels) (ADC_CDR84)	yes	yes
Channel Data Register (External Channels) (ADC_CDR85)	yes	yes
Channel Data Register (External Channels) (ADC_CDR86)	yes	yes
Channel Data Register (External Channels) (ADC_CDR87)	yes	yes
Channel Data Register (External Channels) (ADC_CDR88)	yes	yes
Channel Data Register (External Channels) (ADC_CDR89)	yes	yes
Channel Data Register (External Channels) (ADC_CDR90)	yes	yes
Channel Data Register (External Channels) (ADC_CDR91)	yes	yes
Channel Data Register (External Channels) (ADC_CDR92)	yes	yes
Channel Data Register (External Channels) (ADC_CDR93)	yes	yes
Channel Data Register (External Channels) (ADC_CDR94)	yes	yes
Channel Data Register (External Channels) (ADC_CDR95)	yes	yes
Threshold Register (ADC_THRHLR4)	yes	no
Threshold Register (ADC_THRHLR5)	yes	no
Channel Watchdog Select Register 0 (ADC_CWSELR0)	yes	yes
Channel Watchdog Select Register 1 (ADC_CWSELR1)	yes	yes
Channel Watchdog Select Register 2 (ADC_CWSELR2)	yes	no
Channel Watchdog Select Register 3 (ADC_CWSELR3)	no	no
Channel Watchdog Select Register 4 (ADC_CWSELR4)	yes	yes
Channel Watchdog Select Register 5 (ADC_CWSELR5)	yes	yes
Channel Watchdog Select Register 6 (ADC_CWSELR6)	yes	yes

Table continues on the next page...

Table 34-3. SAR ADC control register applicability (continued)

Register	ADC_0	ADC_1
Channel Watchdog Select Register 7 (ADC_CWSELR7)	yes	yes
Channel Watchdog Select Register 8 (ADC_CWSELR8)	yes	yes
Channel Watchdog Select Register 9 (ADC_CWSELR9)	yes	yes
Channel Watchdog Select Register 10 (ADC_CWSELR10)	yes	yes
Channel Watchdog Select Register 11 (ADC_CWSELR11)	yes	yes
Channel Watchdog Enable Register 0 (ADC_CWENR0)	yes	yes
Channel Watchdog Enable Register 1 (ADC_CWENR1)	yes	yes
Channel Watchdog Enable Register 2 (ADC_CWENR2)	yes	yes
Analog Watchdog Out of Range Register (ADC_AWORR0)	yes	yes
Analog Watchdog Out of Range Register (ADC_AWORR1)	yes	yes
Analog Watchdog Out of Range Register (ADC_AWORR2)	yes	yes
Calibration, BIST Control and status Register (ADC_CALBISTREG)	yes	yes
Offset and Gain User Register (ADC_OFSGNUSR)	yes	yes

34.1.5 ADC channel mapping

The following table shows the ADC channel mapping with respect to Precision[P], Standard[S] and External[X] channels. Please refer to the device IO Signal Description Input Multiplexing Tables workbook for input signal descriptions and the mapping of signals to device pins.

Table 34-4. ADC channel mapping

Channel Number	Source of Inputs	
	ADC_0 (10-bit)	ADC_1 (12-bit)
0	—	ADC1_P[4]
1	—	ADC1_P[5]
2	—	ADC1_P[6]
3	ADC0_S[1]	ADC1_P[7]
4	ADC0_S[0]	ADC1_P[8]
5	—	ADC1_P[9]
6	—	ADC1_P[10]
7	ADC0_S[2]	ADC1_P[11]
8	ADC0_S[8]	ADC1_P[12]
9	—	ADC1_P[0]
10	—	ADC1_P[1]
11	ADC0_S[9]	ADC1_P[2]
12	ADC0_S[10]	ADC1_P[3]

Table continues on the next page...

Table 34-4. ADC channel mapping (continued)

Channel Number	Source of Inputs	
	ADC_0 (10-bit)	ADC_1 (12-bit)
13	—	ADC1_P[13]
14	—	ADC1_P[14]
15	—	ADC1_P[15]
16	—	—
17	PMC_BGAP (PMC bandgap)	—
18	VDD_HV_ADC0	—
19	VSS_HV_ADC0	—
20	VDD_HV_ADC0_REF	—
21	VSS_HV_ADC0_REF	—
22	—	—
23	—	—
24	—	—
25	—	—
26	—	—
27	—	—
28	—	—
29	—	—
30	—	—
31	—	—
32	ADC0_S[11]	ADC1_S[15]
33	ADC0_S[12]	ADC1_S[0]
34	ADC0_S[13]	—
35	ADC0_S[14]	ADC1_S[1]
36	—	ADC1_S[2]
37	—	ADC1_S[3]
38	ADC0_S[15]	ADC1_S[4]
39	ADC0_S[27]	ADC1_S[5]
40	ADC0_S[18]	ADC1_S[6]
41	ADC0_S[17]	ADC1_S[7]
42	ADC0_S[26]	ADC1_S[8]
43	ADC0_S[25]	ADC1_S[9]
44	ADC0_S[31]	ADC1_S[10]
45	ADC0_S[30]	ADC1_S[11]
46	ADC0_S[24]	ADC1_S[12]
47	ADC0_S[23]	ADC1_S[13]
48	ADC0_S[29]	—
49	ADC0_S[28]	PMC_BGAP (PMC bandgap)
50	ADC0_S[22]	VDD_HV_ADC1

Table continues on the next page...

Table 34-4. ADC channel mapping (continued)

Channel Number	Source of Inputs	
	ADC_0 (10-bit)	ADC_1 (12-bit)
51	ADC0_S[3]	VSS_HV_ADC1
52	ADC0_S[4]	VDD_HV_ADC1_REF
53	ADC0_S[5]	VSS_HV_ADC1_REF
54	ADC0_S[6]	—
55	ADC0_S[7]	—
56	ADC0_S[16]	—
57	ADC0_S[19]	—
58	ADC0_S[20]	—
59	ADC0_S[21]	—
60	ADC0_S[44]	—
61	ADC0_S[45]	—
62	ADC0_S[46]	—
63	ADC0_S[47]	—
64	ADC0_X[0]	ADC1_X[0]
65	ADC0_X[0]	ADC1_X[0]
66	ADC0_X[0]	ADC1_X[0]
67	ADC0_X[0]	ADC1_X[0]
68	ADC0_X[0]	ADC1_X[0]
69	ADC0_X[0]	ADC1_X[0]
70	ADC0_X[0]	ADC1_X[0]
71	ADC0_X[0]	ADC1_X[0]
72	ADC0_X[1]	ADC1_X[1]
73	ADC0_X[1]	ADC1_X[1]
74	ADC0_X[1]	ADC1_X[1]
75	ADC0_X[1]	ADC1_X[1]
76	ADC0_X[1]	ADC1_X[1]
77	ADC0_X[1]	ADC1_X[1]
78	ADC0_X[1]	ADC1_X[1]
79	ADC0_X[1]	ADC1_X[1]
80	ADC0_X[2]	ADC1_X[2]
81	ADC0_X[2]	ADC1_X[2]
82	ADC0_X[2]	ADC1_X[2]
83	ADC0_X[2]	ADC1_X[2]
84	ADC0_X[2]	ADC1_X[2]
85	ADC0_X[2]	ADC1_X[2]
86	ADC0_X[2]	ADC1_X[2]
87	ADC0_X[2]	ADC1_X[2]
88	ADC0_X[3]	ADC1_X[3]

Table continues on the next page...

Table 34-4. ADC channel mapping (continued)

Channel Number	Source of Inputs	
	ADC_0 (10-bit)	ADC_1 (12-bit)
89	ADC0_X[3]	ADC1_X[3]
90	ADC0_X[3]	ADC1_X[3]
91	ADC0_X[3]	ADC1_X[3]
92	ADC0_X[3]	ADC1_X[3]
93	ADC0_X[3]	ADC1_X[3]
94	ADC0_X[3]	ADC1_X[3]
95	ADC0_X[3]	ADC1_X[3]

NOTE

For each channel of ADC, the corresponding SIUL_MSCR[APC] should be enabled. For more details, refer SIUL2_MSCRn[APC] and IO signal table that is attached with this document.

34.1.6 ADC_0 valid control register bits

Table 34-5. ADC_0 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_0_MCR	R	OWREN	WLSIDE	MODE	0	0	0	0	NSTART	0	JTRGEN	JEDGE	JSTART	0	0	CTUEN	CTU_MODE
	W																
	R	0	0	0	0	0	0	0	ADCLKSEL	ABORT_CHAIN	ABORT	ACKO	0	0	REFSEL		PWDN
	W																
ADC_0_MSR	R	CALIBRTD	0	0	0	0	0	0	NSTART	JABORT	0	0	JSTART	0	0	0	CTUSTART
	W																
	R	CHADDR							0	0	0	ACKO	0	0	ADCSTATUS		
	W																
ADC_0_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 34-5. ADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	EOCTU	JEOC	JECH	EOC	ECH
	W												w1c	w1c	w1c	w1c	w1c
ADC_0_CEOCFR0	R	0	0	0	0	0	0	0	0	0	0	EOCCH21	EOCCH20	EOCCH19	EOCCH18	0	0
	W											w1c	w1c	w1c	w1c		
	R	0	0	0	EOCCH12	EOCCH11	0	0	EOCCH8	EOCCH7	0	0	EOCCH4	EOCCH3	0	0	0
	W				w1c	w1c			w1c	w1c			w1c	w1c			
ADC_0_CEOCFR1	R	EOCCH63	EOCCH62	EOCCH61	EOCCH60	EOCCH59	EOCCH58	EOCCH57	EOCCH56	EOCCH55	EOCCH54	EOCCH53	EOCCH52	EOCCH51	EOCCH50	EOCCH49	EOCCH48
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	EOCCH47	EOCCH46	EOCCH45	EOCCH44	EOCCH43	EOCCH42	EOCCH41	EOCCH40	EOCCH39	EOCCH38	0	0	EOCCH35	EOCCH34	EOCCH33	EOCCH32
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c
ADC_0_CEOCFR2	R	EOCCH95	EOCCH94	EOCCH93	EOCCH92	EOCCH91	EOCCH90	EOCCH89	EOCCH88	EOCCH87	EOCCH86	EOCCH85	EOCCH84	EOCCH83	EOCCH82	EOCCH81	EOCCH80
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	EOCCH79	EOCCH78	EOCCH77	EOCCH76	EOCCH75	EOCCH74	EOCCH73	EOCCH72	EOCCH71	EOCCH70	EOCCH69	EOCCH68	EOCCH67	EOCCH66	EOCCH65	EOCCH64
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_0_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	Reserved	0	0	0	0	0	0	0	0	0	0	MSKEOCTU	MSKJEOC	MSKJECH	MSKEOC	MSKECH
	W																
ADC_0_CIMR0	R	0	0	0	0	0	0	0	0	0	0	CIM21	CIM20	CIM19	CIM18	0	0

Table continues on the next page...

Table 34-5. ADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W											w1c	w1c	w1c	w1c		
	R	0	0	0	CIM12	CIM11	0	0	CIM8	CIM7	0	0	CIM4	CIM3	0	0	0
	W				w1c	w1c			w1c	w1c			w1c	w1c			
ADC_0_CIMR1	R	CIM63	CIM62	CIM61	CIM60	CIM59	CIM58	CIM57	CIM56	CIM55	CIM54	CIM53	CIM52	CIM51	CIM50	CIM49	CIM48
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	CIM47	CIM46	CIM45	CIM44	CIM43	CIM42	CIM41	CIM40	CIM39	CIM38	0	0	CIM35	CIM34	CIM33	CIM32
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c
ADC_0_CIMR2	R	CIM95	CIM94	CIM93	CIM92	CIM91	CIM90	CIM89	CIM88	CIM87	CIM86	CIM85	CIM84	CIM83	CIM82	CIM81	CIM80
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	CIM79	CIM78	CIM77	CIM76	CIM75	CIM74	CIM73	CIM72	CIM71	CIM70	CIM69	CIM68	CIM67	CIM66	CIM65	CIM64
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_0_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	WDG5H	WDG5L	WDG4H	WDG4L	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W					w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_0_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	MSKWDG5H	MSKWDG5L	MSKWDG4H	MSKWDG4L	MSKWDG3H	MSKWDG3L	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W					w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_0_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
ADC_0_DMAR0	R	0	0	0	0	0	0	0	0	0	0	DMA21	DMA20	DMA19	DMA18	0	0
	W																

Table continues on the next page...

Table 34-5. ADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	DMA12	DMA11	0	0	DMA8	DMA7	0	0	DMA4	DMA3	0	0	0
	W																
ADC_0_DMAR1	R	DMA63	DMA62	DMA61	DMA60	DMA59	DMA58	DMA57	DMA56	DMA55	DMA54	DMA53	DMA52	DMA51	DMA50	DMA49	DMA48
	W																
	R	DMA47	DMA46	DMA45	DMA44	DMA43	DMA42	DMA41	DMA40	DMA39	DMA38	0	0	DMA35	DMA34	DMA33	DMA32
	W																
ADC_0_DMAR2	R	DMA95	DMA94	DMA93	DMA92	DMA91	DMA90	DMA89	DMA88	DMA87	DMA86	DMA85	DMA84	DMA83	DMA82	DMA81	DMA80
	W																
	R	DMA79	DMA78	DMA77	DMA76	DMA75	DMA74	DMA73	DMA72	DMA71	DMA70	DMA69	DMA68	DMA67	DMA66	DMA65	DMA64
	W																
ADC_0_THRHLR0-3	R	0	0	0	0	0	0	THRH									
	W																
	R	0	0	0	0	0	0	THRL									
	W																
ADC_0_PSCR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	PREVAL2		PREVAL1		PREVAL0		PRECONV
	W																
ADC_0_PSR0	R	0	0	0	0	0	0	0	0	0	0	PRES21	PRES20	PRES19	PRES18	0	0
	W																
	R	0	0	0	PRES12	PRES11	0	0	PRES8	PRES7	0	0	PRES4	PRES3	0	0	0
	W																
ADC_0_PSR1	R	PRES63	PRES62	PRES61	PRES60	PRES59	PRES58	PRES57	PRES56	PRES55	PRES54	PRES53	PRES52	PRES51	PRES50	PRES49	PRES48
	W																
	R	PRES47	PRES46	PRES45	PRES44	PRES43	PRES42	PRES41	PRES40	PRES39	PRES38	0	0	PRES35	PRES34	PRES33	PRES32
	R																
ADC_0_PSR2	R	PRES95	PRES94	PRES93	PRES92	PRES91	PRES90	PRES89	PRES88	PRES87	PRES86	PRES85	PRES84	PRES83	PRES82	PRES81	PRES80
	W																
	R	PRES79	PRES78	PRES77	PRES76	PRES75	PRES74	PRES73	PRES72	PRES71	PRES70	PRES69	PRES68	PRES67	PRES66	PRES65	PRES64
	R																
ADC_0_CTR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 34-5. ADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
	R	0	0	0	0	0	0	0	0	INPSAMP							
	W																
ADC_0_CTR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	INPSAMP							
	W																
ADC_0_CTR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	INPSAMP							
	W																
ADC_0_NCMR0	R	0	0	0	0	0	0	0	0	0	0	CH21	CH20	CH19	CH18	0	0
	W																
	R	0	0	0	CH12	CH11	0	0	CH8	CH7	0	0	CH4	CH3	0	0	0
	W																
ADC_0_NCMR1	R	CH63	CH62	CH61	CH60	CH59	CH58	CH57	CH56	CH55	CH54	CH53	CH52	CH51	CH50	CH49	CH48
	W																
	R	CH47	CH46	CH45	CH44	CH43	CH42	CH41	CH40	CH39	CH38	0	0	CH35	CH34	CH33	CH32
	W																
ADC_0_NCMR2	R	CH95	CH94	CH93	CH92	CH91	CH90	CH89	CH88	CH87	CH86	CH85	CH84	CH83	CH82	CH81	CH80
	W																
	R	CH79	CH78	CH77	CH76	CH75	CH74	CH73	CH72	CH71	CH70	CH69	CH68	CH67	CH66	CH65	CH64
	W																
ADC_0_JCMR0	R	0	0	0	0	0	0	0	0	0	0	CH21	CH20	CH19	CH18	0	0
	W																
	R	0	0	0	CH12	CH11	0	0	CH8	CH7	0	0	CH4	CH3	0	0	0
	W																
ADC_0_JCMR1	R	CH63	CH62	CH61	CH60	CH59	CH58	CH57	CH56	CH55	CH54	CH53	CH52	CH51	CH50	CH49	CH48
	W																
	R	CH47	CH46	CH45	CH44	CH43	CH42	CH41	CH40	CH39	CH38	0	0	CH35	CH34	CH33	CH32
	W																
ADC_0_JCMR2	R	CH95	CH94	CH93	CH92	CH91	CH90	CH89	CH88	CH87	CH86	CH85	CH84	CH83	CH82	CH81	CH80
	W																
	R	CH79	CH78	CH77	CH76	CH75	CH74	CH73	CH72	CH71	CH70	CH69	CH68	CH67	CH66	CH65	CH64
	W																
ADC_0_DSDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	DSD															

Table continues on the next page...

Table 34-5. ADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
ADC_0_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
ADC_0_CDR*n*	R	0	0	0	0	0	0	0	0	0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	CDATA															
	W																
ADC_0_THRHLR4-5	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_0_CWSELR0	R	WSEL_CH7				0				0				WSEL_CH4			
	W																
	R	WSEL_CH3				0				0				0			
	W																
ADC_0_CWSELR1	R	0				0				0				WSEL_CH12			
	W																
	R	WSEL_CH11				0				0				WSEL_CH8			
	W																
ADC_0_CWSELR2	R	0				0				WSEL_CH21				WSEL_CH20			
	W																
	R	WSEL_CH19				WSEL_CH18				0				0			
	W																
ADC_0_CWSELR3	R	0				0				0				0			
	W																
	R	0				0				0				0			
	W																
ADC_0_CWSELR4	R	WSEL_CH39				WSEL_CH38				0				0			
	W																
	R	WSEL_CH35				WSEL_CH34				WSEL_CH33				WSEL_CH32			
	W																
ADC_0_CWSELR5	R	WSEL_CH47				WSEL_CH46				WSEL_CH45				WSEL_CH44			
	W																
	R	WSEL_CH43				WSEL_CH42				WSEL_CH41				WSEL_CH40			
	W																

Table continues on the next page...

Table 34-5. ADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_0_CWSELR6	R	WSEL_CH55				WSEL_CH54				WSEL_CH53				WSEL_CH52			
	W																
	R	WSEL_CH51				WSEL_CH50				WSEL_CH49				WSEL_CH48			
	W																
ADC_0_CWSELR7	R	WSEL_CH63				WSEL_CH62				WSEL_CH61				WSEL_CH60			
	W																
	R	WSEL_CH59				WSEL_CH58				WSEL_CH57				WSEL_CH56			
	W																
ADC_0_CWSELR8	R	WSEL_CH71				WSEL_CH70				WSEL_CH69				WSEL_CH68			
	W																
	R	WSEL_CH67				WSEL_CH66				WSEL_CH65				WSEL_CH64			
	W																
ADC_0_CWSELR9	R	WSEL_CH79				WSEL_CH78				WSEL_CH77				WSEL_CH76			
	W																
	R	WSEL_CH75				WSEL_CH74				WSEL_CH73				WSEL_CH72			
	W																
ADC_0_CWSELR10	R	WSEL_CH87				WSEL_CH86				WSEL_CH85				WSEL_CH84			
	W																
	R	WSEL_CH83				WSEL_CH82				WSEL_CH81				WSEL_CH80			
	W																
ADC_0_CWSELR10	R	WSEL_CH95				WSEL_CH94				WSEL_CH93				WSEL_CH92			
	W																
	R	WSEL_CH91				WSEL_CH90				WSEL_CH89				WSEL_CH88			
	W																
ADC_0_CWENR0	R	0	0	0	0	0	0	0	0	0	0	CWEN21	CWEN20	CWEN19	CWEN18	0	0
	W																
	R	0	0	0	CWEN12	CWEN11	0	0	CWEN8	CWEN7	0	0	CWEN4	CWEN3	0	0	0
	W																
ADC_0_CWENR1	R	CWEN63	CWEN62	CWEN61	CWEN60	CWEN59	CWEN58	CWEN57	CWEN56	CWEN55	CWEN54	CWEN53	CWEN52	CWEN51	CWEN50	CWEN49	CWEN48
	W																
	R	CWEN47	CWEN46	CWEN45	CWEN44	CWEN43	CWEN42	CWEN41	CWEN40	CWEN39	CWEN38	0	0	CWEN35	CWEN34	CWEN33	CWEN32
	W																
ADC_0_CWENR2	R	CWEN95	CWEN94	CWEN93	CWEN92	CWEN91	CWEN90	CWEN89	CWEN88	CWEN87	CWEN86	CWEN85	CWEN84	CWEN83	CWEN82	CWEN81	CWEN80
	W																

Table continues on the next page...

Table 34-5. ADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	CWEN79	CWEN78	CWEN77	CWEN76	CWEN75	CWEN74	CWEN73	CWEN72	CWEN71	CWEN70	CWEN69	CWEN68	CWEN67	CWEN66	CWEN65	CWEN64
	W																
ADC_0_AWORR0	R	0	0	0	0	0	0	0	0	0	0	CH21	CH20	CH19	CH18	0	0
	W											w1c	w1c	w1c	w1c		
	R	0	0	0	AWOR_CH12	AWOR_CH11	0	0	AWOR_CH8	AWOR_CH7	0	0	AWOR_CH4	AWOR_CH3	0	0	0
	W				w1c	w1c			w1c	w1c			w1c	w1c			
ADC_0_AWORR1	R	AWOR_CH63	AWOR_CH62	AWOR_CH61	AWOR_CH60	AWOR_CH59	AWOR_CH58	AWOR_CH57	AWOR_CH56	AWOR_CH55	AWOR_CH54	AWOR_CH53	AWOR_CH52	AWOR_CH51	AWOR_CH50	AWOR_CH49	AWOR_CH48
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	AWOR_CH47	AWOR_CH46	AWOR_CH45	AWOR_CH44	AWOR_CH43	AWOR_CH42	AWOR_CH41	AWOR_CH40	AWOR_CH39	AWOR_CH38	0	0	AWOR_CH35	AWOR_CH34	AWOR_CH33	AWOR_CH32
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c
ADC_0_AWORR2	R	AWOR_CH95	AWOR_CH94	AWOR_CH93	AWOR_CH92	AWOR_CH91	AWOR_CH90	AWOR_CH89	AWOR_CH88	AWOR_CH87	AWOR_CH86	AWOR_CH85	AWOR_CH84	AWOR_CH83	AWOR_CH82	AWOR_CH81	AWOR_CH80
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	AWOR_CH79	AWOR_CH78	AWOR_CH77	AWOR_CH76	AWOR_CH75	AWOR_CH74	AWOR_CH73	AWOR_CH72	AWOR_CH71	AWOR_CH70	AWOR_CH69	AWOR_CH68	AWOR_CH67	AWOR_CH66	AWOR_CH65	AWOR_CH64
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_0_STCR1	R	INPSAMP_C								Reserved							
	W																
	R	INPSAMP_S								Reserved							
	W																

Table continues on the next page...

Table 34-5. ADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_0_STCR2	R	0	0	0	0	MSKWDSERR	SERR	MSKWDTERR	0	MSKST_EOC	0	0	0	0	MSKWDG_EOA_C	0	MSKWDG_EOA_S
	W																
	R	MSKERR_C	0	MSKERR_S2	MSKERR_S1	MSKERR_S0	0	0	0	MSKWDG_EOA_C	0	0	FMA_WDSERR	FMA_WDTERR	FMA_C	FMA_RC	FMA_S
	W																
ADC_0_STCR3	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	ALG		0	0	0	MSTEP				
	W																
ADC_0_STBRR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	WDT		
	W																
	R	0	0	0	0	0	0	0	0	BR							
	W																
ADC_0_STSR1	R	0	0	0	0	WDSERR	0	WDTERR	OVERWR	ST_EOC	0	0	0	0	WDG_EOA_C	0	WDG_EOA_S
	W					w1c		w1c	w1c	w1c					w1c		w1c
	R	ERR_C	0	ERR_S2	ERR_S1	ERR_S0	0	STEP_C					0	0	0	0	0
	W	w1c		w1c	w1c	w1c											
ADC_0_STSR2	R	OVFL	0	0	0	DATA1											
	W																
	R	0	0	0	0	DATA0											
	W																
ADC_0_STSR3	R	0	0	0	0	DATA1											
	W																
	R	0	0	0	0	DATA0											
	W																
ADC_0_STSR4	R	0	0	0	0	DATA1											
	W																
	R	0	0	0	0	DATA0											

Table continues on the next page...

Table 34-5. ADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
ADC_0_STDR1	R	0	0	0	0	0	0	0	0	0	0	0	0	VALID	OWERWR	0	0
	W																
	R	0	0	0	0	TCDATA											
	W																
ADC_0_STDR2	R	FDATA												VALID	OWERWR	0	0
	W																
	R	0	0	0	0	IDATA											
	W																
ADC_0_STAW0R	R	AWDE	WDTE	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_0_STAW1AR	R	AWDE	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_0_STAW1BR	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_0_STAW2R	R	AWDE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	THRL											
	W																
ADC_0_STAW4R	R	AWDE	WDTE	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_0_STAW5R	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																

Table continues on the next page...

Table 34-5. ADC_0 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_0_CALBISTREG	R	Reserved	Reserved		TSAMP		0	0	0	Reserved							
	W																
	R	C_T_BUSY	0	Reserved						0	NR_SMPL		AVG_EN	TEST_FAIL	0	0	TEST_EN
	W													w1c			
ADC_0_OFSGNUSR	R	0	0	0	0	0	0	GAIN_USER									
	W																
	R	0	0	0	0	0	0	OFFSET_USER									
	W																
ADC_0_ALTR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

34.1.7 ADC_1 valid control register bits

Table 34-6. ADC_1 register definitions

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_1_MCR	R	OWREN	WLSIDE	MODE	0	0	0	0	NSTART	0	JTRGEN	JEDGE	JSTART	0	0	CTUEN	CTU_MODE
	W																
	R	0	0	0	0	0	0	0	ADCLKSEL	ABORT_CHAIN	ABORT	ACKO	0	0	REFSEL		PWDN
	W																
ADC_1_MSR	R	CALIBRTD	0	0	0	0	0	0	NSTART	JABORT	0	0	JSTART	0	0	0	CTUSTART
	W																

Table continues on the next page...

Table 34-6. ADC_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	CHADDR							0	0	0	ACKO	0	0	ADCSTATUS		
	W																
ADC_1_ISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	EOCTU	JEOC	JECH	EOC	ECH
	W												w1c	w1c	w1c	w1c	w1c
ADC_1_CEOCFR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	EOCCH15	EOCCH14	EOCCH13	EOCCH12	EOCCH11	EOCCH10	EOCCH9	EOCCH8	EOCCH7	EOCCH6	EOCCH5	EOCCH4	EOCCH3	EOCCH2	EOCCH1	EOCCH0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_1_CEOCFR1	R	0	0	0	0	0	0	0	0	0	0	EOCCH53	EOCCH52	EOCCH51	EOCCH50	0	0
	W											w1c	w1c	w1c	w1c		
	R	EOCCH47	EOCCH46	EOCCH45	EOCCH44	EOCCH43	EOCCH42	EOCCH41	EOCCH40	EOCCH39	EOCCH38	EOCCH37	EOCCH36	EOCCH35	EOCCH34	EOCCH33	EOCCH32
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_1_CEOCFR2	R	EOCCH95	EOCCH94	EOCCH93	EOCCH92	EOCCH91	EOCCH90	EOCCH89	EOCCH88	EOCCH87	EOCCH86	EOCCH85	EOCCH84	EOCCH83	EOCCH82	EOCCH81	EOCCH80
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	EOCCH79	EOCCH78	EOCCH77	EOCCH76	EOCCH75	EOCCH74	EOCCH73	EOCCH72	EOCCH71	EOCCH70	EOCCH69	EOCCH68	EOCCH67	EOCCH66	EOCCH65	EOCCH64
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_1_IMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	Reserved	0	0	0	0	0	0	0	0	0	0	MSKEOCTU	MSKJEOC	MSKJECH	MSKEOC	MSKECH
	W																
ADC_1_CIMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table continues on the next page...

Table 34-6. ADC_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
	R	CIM15	CIM14	CIM13	CIM12	CIM11	CIM10	CIM9	CIM8	CIM7	CIM6	CIM5	CIM4	CIM3	CIM2	CIM1	CIM0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_1_CIMR1	R	0	0	0	0	0	0	0	0	0	0	CIM53	CIM52	CIM51	CIM50	0	0
	W											w1c	w1c	w1c	w1c		
	R	CIM47	CIM46	CIM45	CIM44	CIM43	CIM42	CIM41	CIM40	CIM39	CIM38	CIM37	CIM36	CIM35	CIM34	CIM33	CIM32
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_1_CIMR2	R	CIM95	CIM94	CIM93	CIM92	CIM91	CIM90	CIM89	CIM88	CIM87	CIM86	CIM85	CIM84	CIM83	CIM82	CIM81	CIM80
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	CIM79	CIM78	CIM77	CIM76	CIM75	CIM74	CIM73	CIM72	CIM71	CIM70	CIM69	CIM68	CIM67	CIM66	CIM65	CIM64
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_1_WTISR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
	W											w1c	w1c	w1c	w1c	w1c	w1c
ADC_1_WTIMR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	MSKWDG2H	MSKWDG2L	MSKWDG1H	MSKWDG1L	MSKWDG0H	MSKWDG0L
	W											w1c	w1c	w1c	w1c	w1c	w1c
ADC_1_DMAE	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DCLR	DMAEN
	W																
ADC_1_DMAR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 34-6. ADC_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	DMA15	DMA14	DMA13	DMA12	DMA11	DMA10	DMA9	DMA8	DMA7	DMA6	DMA5	DMA4	DMA3	DMA2	DMA1	DMA0
	W																
ADC_1_DMAR1	R	0	0	0	0	0	0	0	0	0	0	DMA53	DMA52	DMA51	DMA50	0	0
	W																
	R	DMA47	DMA46	DMA45	DMA44	DMA43	DMA42	DMA41	DMA40	DMA39	DMA38	DMA37	DMA36	DMA35	DMA34	DMA33	DMA32
	W																
ADC_1_DMAR2	R	DMA95	DMA94	DMA93	DMA92	DMA91	DMA90	DMA89	DMA88	DMA87	DMA86	DMA85	DMA84	DMA83	DMA82	DMA81	DMA80
	W																
	R	DMA79	DMA78	DMA77	DMA76	DMA75	DMA74	DMA73	DMA72	DMA71	DMA70	DMA69	DMA68	DMA67	DMA66	DMA65	DMA64
	W																
ADC_1_THRHLR0-2	R	0	0	0	0	THRH											
	W																
	R	0	0	0	0	THRL											
	W																
ADC_1_PSCR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	PREVAL2		PREVAL1		PREVAL2		PRECONV
	W																
ADC_1_PSR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	PRES15	PRES14	PRES13	PRES12	PRES11	PRES10	PRES9	PRES8	PRES7	PRES6	PRES5	PRES4	PRES3	PRES2	PRES1	PRES0
	W																
ADC_1_PSR1	R	0	0	0	0	0	0	0	0	0	0	PRES53	PRES52	PRES51	PRES50	0	0
	W																
	R	PRES47	PRES46	PRES45	PRES44	PRES43	PRES42	PRES41	PRES40	PRES39	PRES38	PRES37	PRES36	PRES35	PRES34	PRES33	PRES32
	W																
ADC_1_PSR2	R	PRES95	PRES94	PRES93	PRES92	PRES91	PRES90	PRES89	PRES88	PRES87	PRES86	PRES85	PRES84	PRES83	PRES82	PRES81	PRES80
	W																
	R	PRES79	PRES78	PRES77	PRES76	PRES75	PRES74	PRES73	PRES72	PRES71	PRES70	PRES69	PRES68	PRES67	PRES66	PRES65	PRES64
	W																
ADC_1_CTR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

Table continues on the next page...

Table 34-6. ADC_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	R	0	0	0	0	0	0	0	0	INPSAMP							
	W																
ADC_1_CTR1	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	INPSAMP							
	W																
ADC_1_CTR2	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	INPSAMP							
	W																
ADC_1_NCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R W	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
ADC_1_NCMR1	R	0	0	0	0	0	0	0	0	0	0	CH53	CH52	CH51	CH50	0	0
	W																
	R W	CH47	CH46	CH45	CH44	CH43	CH42	CH41	CH40	CH39	CH38	CH37	CH36	CH35	CH34	CH33	CH32
ADC_1_NCMR2	R W	CH95	CH94	CH93	CH92	CH91	CH90	CH89	CH88	CH87	CH86	CH85	CH84	CH83	CH82	CH81	CH80
	R W	CH79	CH78	CH77	CH76	CH75	CH74	CH73	CH72	CH71	CH70	CH69	CH68	CH67	CH66	CH65	CH64
ADC_1_JCMR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R W	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
ADC_1_JCMR1	R	0	0	0	0	0	0	0	0	0	0	CH53	CH52	CH51	CH50	0	0
	W																
	R W	CH47	CH46	CH45	CH44	CH43	CH42	CH41	CH40	CH39	CH38	CH37	CH36	CH35	CH34	CH33	CH32
ADC_1_JCMR2	R W	CH95	CH94	CH93	CH92	CH91	CH90	CH89	CH88	CH87	CH86	CH85	CH84	CH83	CH82	CH81	CH80
	R W	CH79	CH78	CH77	CH76	CH75	CH74	CH73	CH72	CH71	CH70	CH69	CH68	CH67	CH66	CH65	CH64
ADC_1_DSDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R W	DSD															

Table continues on the next page...

Table 34-6. ADC_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_1_PDEDR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	PDED							
	W																
ADC_1_CDR*n*	R	0	0	0	0	0	0	0	0	0	0	0	0	VALID	OVERW	RESULT	
	W																
	R	CDATA															
	W																
ADC_1_CWSELR0	R	WSEL_CH7				WSEL_CH6				WSEL_CH5				WSEL_CH4			
	W																
	R	WSEL_CH3				WSEL_CH2				WSEL_CH1				WSEL_CH0			
	W																
ADC_1_CWSELR1	R	WSEL_CH15				WSEL_CH14				WSEL_CH13				WSEL_CH12			
	W																
	R	WSEL_CH11				WSEL_CH10				WSEL_CH9				WSEL_CH8			
	W																
ADC_1_CWSELR2	R	0				0				0				0			
	W																
	R	0				0				0				0			
	W																
ADC_1_CWSELR3	R	0				0				0				0			
	W																
	R	0				0				0				0			
	W																
ADC_1_CWSELR4	R	WSEL_CH39				WSEL_CH38				WSEL_CH37				WSEL_CH36			
	W																
	R	WSEL_CH35				WSEL_CH34				WSEL_CH33				WSEL_CH32			
	W																
ADC_1_CWSELR5	R	WSEL_CH47				WSEL_CH46				WSEL_CH45				WSEL_CH44			
	W																
	R	WSEL_CH43				WSEL_CH42				WSEL_CH41				WSEL_CH40			
	W																
ADC_1_CWSELR6	R	0				0				WSEL_CH53				WSEL_CH52			
	W																
	R	WSEL_CH51				WSEL_CH50				0				0			
	W																
ADC_1_CWSELR8	R	WSEL_CH71				WSEL_CH70				WSEL_CH69				WSEL_CH68			

Table continues on the next page...

Table 34-6. ADC_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	W																
	R	WSEL_CH67				WSEL_CH66				WSEL_CH65				WSEL_CH64			
	W																
ADC_1_CWSELR9	R	WSEL_CH79				WSEL_CH78				WSEL_CH77				WSEL_CH76			
	W																
	R	WSEL_CH75				WSEL_CH74				WSEL_CH73				WSEL_CH72			
	W																
ADC_1_CWSELR10	R	WSEL_CH87				WSEL_CH86				WSEL_CH85				WSEL_CH84			
	W																
	R	WSEL_CH83				WSEL_CH82				WSEL_CH81				WSEL_CH80			
	W																
ADC_1_CWSELR10	R	WSEL_CH95				WSEL_CH94				WSEL_CH93				WSEL_CH92			
	W																
	R	WSEL_CH91				WSEL_CH90				WSEL_CH89				WSEL_CH88			
	W																
ADC_1_CWENR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	CWEN15	CWEN14	CWEN13	CWEN12	CWEN11	CWEN10	CWEN9	CWEN8	CWEN7	CWEN6	CWEN5	CWEN4	CWEN3	CWEN2	CWEN1	CWEN0
	W																
ADC_1_CWENR1	R	0	0	0	0	0	0	0	0	0	0	CWEN53	CWEN52	CWEN51	CWEN50	0	0
	W																
	R	CWEN47	CWEN46	CWEN45	CWEN44	CWEN43	CWEN42	CWEN41	CWEN40	CWEN39	CWEN38	CWEN37	CWEN36	CWEN35	CWEN34	CWEN33	CWEN32
	W																
ADC_1_CWENR2	R	CWEN95	CWEN94	CWEN93	CWEN92	CWEN91	CWEN90	CWEN89	CWEN88	CWEN87	CWEN86	CWEN85	CWEN84	CWEN83	CWEN82	CWEN81	CWEN80
	W																
	R	CWEN79	CWEN78	CWEN77	CWEN76	CWEN75	CWEN74	CWEN73	CWEN72	CWEN71	CWEN70	CWEN69	CWEN68	CWEN67	CWEN66	CWEN65	CWEN64
	W																
ADC_1_AWORR0	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	AWOR_CH15	AWOR_CH14	AWOR_CH13	AWOR_CH12	AWOR_CH11	AWOR_CH10	AWOR_CH9	AWOR_CH8	AWOR_CH7	AWOR_CH6	AWOR_CH5	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c

Table continues on the next page...

Table 34-6. ADC_1 register definitions (continued)

Name		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADC_1_AWORR1	R	0	0	0	0	0	0	0	0	0	0	AWOR_CH53	AWOR_CH52	AWOR_CH51	AWOR_CH50	0	0
	W											w1c	w1c	w1c	w1c		
	R	AWOR_CH47	AWOR_CH46	AWOR_CH45	AWOR_CH44	AWOR_CH43	AWOR_CH42	AWOR_CH41	AWOR_CH40	AWOR_CH39	AWOR_CH38	AWOR_CH37	AWOR_CH36	AWOR_CH35	AWOR_CH34	AWOR_CH33	AWOR_CH32
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_1_AWORR2	R	AWOR_CH95	AWOR_CH94	AWOR_CH93	AWOR_CH92	AWOR_CH91	AWOR_CH90	AWOR_CH89	AWOR_CH88	AWOR_CH87	AWOR_CH86	AWOR_CH85	AWOR_CH84	AWOR_CH83	AWOR_CH82	AWOR_CH81	AWOR_CH80
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
	R	AWOR_CH79	AWOR_CH78	AWOR_CH77	AWOR_CH76	AWOR_CH75	AWOR_CH74	AWOR_CH73	AWOR_CH72	AWOR_CH71	AWOR_CH70	AWOR_CH69	AWOR_CH68	AWOR_CH67	AWOR_CH66	AWOR_CH65	AWOR_CH64
	W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
ADC_1_CALBISTREG	R	Reserved	Reserved		TSAMP		0	0	0	Reserved							
	W																
	R	C_T_BUSY	0	Reserved						0	NR_SMPL		AVG_EN	TEST_FAIL	0	0	TEST_EN
	W													w1c			
ADC_1_OFSGNUSR	R	0	0	0	0	0	0	GAIN_USER									
	W																
	R	0	0	0	0	0	0	OFFSET_USER									
	W																
ADC_1_ALTR	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																
	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	W																

34.1.8 BCTU Interface

Both ADCs are triggered from the BCTU. The BCTU provides channel conversion commands to the ADC (includes channel number information). In addition, the ADC provides the conversion result back to the BCTU.

34.1.9 DMA interface

Each ADC supports a single DMA request that can be requested after the conversion of every channel

NOTE

The last ongoing conversion will be aborted as soon as ADC gets STOP request from the software and it will move to STOP mode.

34.1.10 ADC's stop-stop ack protocol

The software must ensure to poll ADC_MSR[ADCSTATUS] to be "idle" before asserting the stop request to the ADC module.

34.2 Introduction

The analog-to-digital converter (ADC) uses Successive Approximation Register (SAR) architecture and provides accurate and fast conversions for a wide range of applications.

34.3 Overview

34.3.1 Parts of the ADC

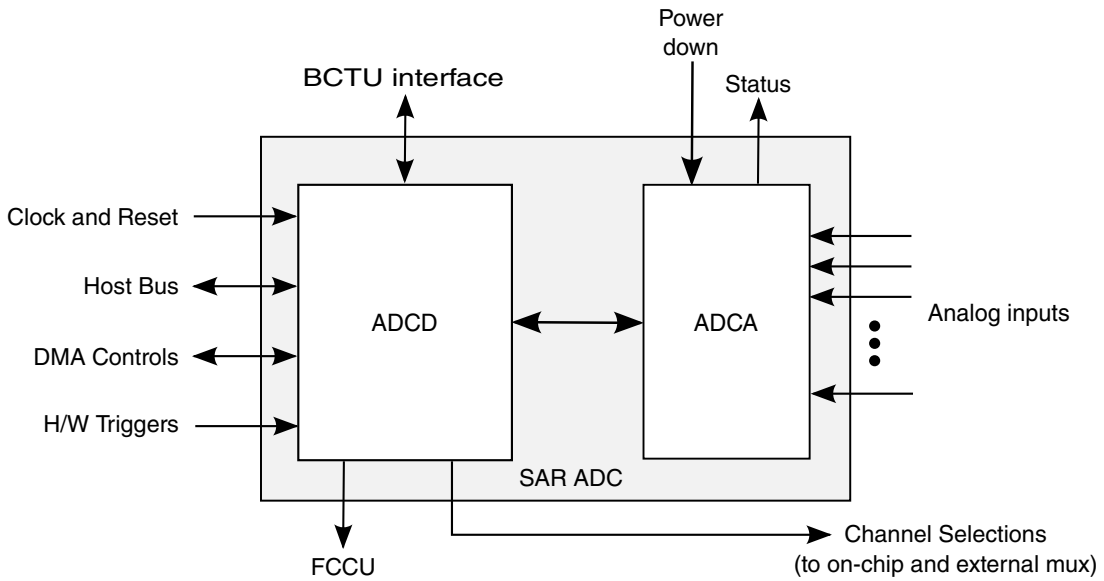


Figure 34-2. ADC high-level block diagram

The ADC is comprised of:

- An ADC digital interface (ADCD). The ADCD holds the configuration, control, and status registers accessible to software via the host bus. The converted value is stored in a register and then transferred through the DMA or host access.
- An ADC analog hard macrocell (ADCA). The analog channel inputs are fed to the inputs of the ADCA. Each channel is sampled for a specific duration and compared with an analog voltage generated with digital code via a digital-to-analog converter (DAC) in the ADCA. The comparison result is given to the ADCD to generate the converted digital value.

34.3.2 ADC clock signals

The ADC uses the following clock signals:

- Bus clock—bus clock at ADCD. Used to access the registers and the operating clock of inter-module interfaces.
- AD_clk—derived from *bus clock*, this signal serves as the operating clock for the ADCA and the SAR controller inside the ADCD.

The relation between the two clocks is determined by the setting of the ADC_MCR[ADCLKSEL] field.

- If ADC_MCR[ADCLKSEL] is 1, the clock signals are the same
- If ADC_MCR[ADCLKSEL] is 0, AD_clk is half of the bus clock.

NOTE

1. Regardless of the setting of the ADC_MCR[ADCLKSEL] field, the maximum clock speed for the ADC clock (AD_clk) is 80 MHz.
2. Depending on your chip's configuration and timing needs, MCR[ADCLKSEL] may always need to be 1.

If the bus clock for the microcontroller is 80 MHz and ADC_MCR[ADCLKSEL] is 1, the ADC clock (AD_clk) is 80 MHz.

If the bus clock for the microcontroller is 80 MHz and ADC_MCR[ADCLKSEL] is 0, the ADC clock (AD_clk) is 40 MHz.

See ADCLKSEL for details on the ADC_MCR register fields.

34.3.3 About ADC channels

ADC channels are divided into 3 groups, each with independent configuration settings. Each group may have different accuracy/performance levels depending on system need (check device configuration for details).

The groups are:

- Group 1, referred to as *precision* channels:
  - 0 - 23 (ADC0—10 bit)
  - 0 - 15 (ADC1—12 bit)
- Group 2, referred to as *internal* channels:
  - 32 - 63 (ADC0—10 bit)
  - 32 - 57 (ADC1—12 bit)
- Group 3, referred to as *external* channels[1] :
  - 64 to 95 (ADC0—10 bit)
  - 64 to 95 (ADC1—12 bit)

NOTE

Accuracy of *external* channels can be increased by increasing the sampling duration, i.e., by increasing the value in the ADC_CTR*n*[INPSAMP] register field. This should be done with caution, though, because increasing sampling duration also

1. This group has an 8 x 1 external multiplexing option. A 3-bit output control is provided for shared use by 4 multiplexers.

increases total conversion time, resulting in a *decrease* in the number of samples per second.

34.3.3.1 Channel configuration

ADC channels can be configured in the following ways:

- The sampling duration of each channel type, i.e., *precision* channels, *internal* channels, *external* channels, is controlled independently, through programming the respective Conversion Timing Register (CTR).
- Separate mask registers for each channel type can be programmed to configure the channels to be converted (each bit in these registers represents one channel).
- External channel selection is provided through external decode signals and is available at the port. Each set of eight external channels can be mapped to an analog input. (See the chip-specific configuration for availability.)
- Analog watchdogs allow continuous hardware monitoring of the converted value from input channels. (See the chip-specific configuration for availability.)
- Presampling

NOTE

Each type of channel is converted in the same way.

34.3.3.2 Initiating conversions

Conversions can be initiated by either software or hardware.

The ADCD has an on-chip Cross-Triggering Unit (BCTU) interface that can also initiate conversions. The BCTU interface supports BCTU Trigger mode and BCTU Control mode (mode is selected via setting the ADC_MCR[CTU_MODE] field).

An example of an application for which BCTU triggering is suitable is a current control loop for lamps or LEDs. Using an on-chip timer module as trigger source, a BCTU can initiate periodic conversions.

34.3.3.3 Interrupt/DMA support

The ADCD provides interrupt/DMA support for each type of channel for various end-of-channel conversion conditions. Data can be transferred via DMA.

34.4 Features

The following general features are implemented in the ADC. (See the chip-specific section for availability of features, as each instance of an ADC within a chip may support different features.)

- Resolution: 10- or 12-bit
- Maximum speed: 1 Mega samples/second at 80 MHz
- Up to 88 channels
- Separate sampling time register for each available range of channels:
  - *precision* channels:
    - 0 - 23 (ADC0—10 bit)
    - 0 - 15 (ADC1—12 bit)
  - *internal* channels:
    - 32 - 63 (ADC0—10 bit)
    - 32 - 57 (ADC1—12 bit)
  - *external* channels:
    - 64 to 95 (ADC0—10 bit)
    - 64 to 95 (ADC1—12 bit)
- External decode signals available for (8 × 1) configuration (external channels only)
- Modes of operation:
  - Normal
  - Injected
  - BCTU
- Normal mode supports One-Shot/Scan (continuous)
- Injected mode supports One-Shot
- Presampling
- Power-Down mode
- Two abort features that allow you to abort either a single channel in a chain or the full chain in normal and injected mode
- Each channel has a dedicated register for conversion results and mode of operation (normal, injected, or BCTU)
- Analog watchdog support:
  - Up to 6 analog watchdogs
  - Configurable for different channels
- Auto clock-off feature
- Programmable clock prescaler (bus clock divided by 2)
- Two BCTU modes: Control and Trigger
- DMA support for each channel
- Interrupts for the following conditions:

    - End of conversion for a single channel for both normal and injected conversions
    - End of conversion for a chain for both normal and injected conversions
    - End of BCTU conversion
    - Watchdog thresholds crossover
- Software-initiated calibration

34.5 Memory Map/Register Definition

The following tables describe the registers and bit meanings for this module.

NOTE

There will be an Access Error for Reserved and out-of-range addresses. Special exceptions are the Reserved spaces at 039Ch and 03A4h: Do not write or change any value in these locations, or the ADC might malfunction.

NOTE

See the chip-specific configuration information because not all registers are available in all ADC instances.

Control logic registers:

- Main Configuration Register (ADC_MCR)
- Main Status register (ADC_MSR)
- Interrupt Status Register (ADC_ISR)
- Channel Pending register 0 (ADC_CEOCFR0)
- Channel Pending register 1 (ADC_CEOCFR1)
- Channel Pending register 2 (ADC_CEOCFR2)
- Interrupt Mask Register (ADC_IMR)
- Channel Interrupt Mask Register 0 (ADC_CIMR0)
- Channel Interrupt Mask Register 1 (ADC_CIMR1)
- Channel Interrupt Mask Register 2 (ADC_CIMR2)
- Watchdog Threshold Interrupt Status Register (ADC_WTISR)
- Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)
- Analog Watchdog Out of Range Register 0 (ADC_AWORR0)
- Analog Watchdog Out of Range Register 1 (ADC_AWORR1)
- Analog Watchdog Out of Range Register 2 (ADC_AWORR2)

DMA registers:

- DMA Enable register (ADC_DMAE)
- DMA Channel Select Register 0 (ADC_DMAR0)

- DMA Channel Select Register 1 (ADC_DMAR1)
- DMA Channel Select Register 2 (ADC_DMAR2)

Threshold registers:

These registers are used to store the user programmable lower and upper thresholds 12-bit values. These registers include:

- Threshold Register (ADC_THRHLR*n*)

Presampling registers:

- Presampling Control Register (ADC_PSCR)
- Presampling register 0 (ADC_PSR0)
- Presampling register 1 (ADC_PSR1)
- Presampling register 2 (ADC_PSR2)

Conversion timing registers:

- Conversion Timing Register 0 (ADC_CTR0)
- Conversion Timing Register 1 (ADC_CTR1)
- Conversion Timing Register 2 (ADC_CTR2)

Mask register: These registers are used to program which input channels must be converted during Normal and Injected conversion. These registers include:

- Normal Conversion Mask Register 0 (ADC_NCMR0)
- Normal Conversion Mask Register 1 (ADC_NCMR1)
- Normal Conversion Mask Register 2 (ADC_NCMR2)
- Injected Conversion Mask Register 0 (ADC_JCMR0)
- Injected Conversion Mask Register 1 (ADC_JCMR1)
- Injected Conversion Mask Register 2 (ADC_JCMR2)

Delay registers:

- Power Down Exit Delay Register (ADC_PDEDR)
- Decode Signals Delay Register (ADC_DSDR)

Data registers:

The conversion results for the internal channels are loaded into data registers. Each data register also gives information regarding the corresponding result. These registers start with CDR0:

- Channel Data Register *n* (Precision Channels) (ADC_CDR*n*)

Other registers include:

- Calibration, BIST Control and status Register (ADC_CALBISTREG)
- Offset and Gain User Register (ADC_OFSGNUSR)

ADC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Main Configuration Register (ADC_MCR)	32	R/W	0000_0101h	34.5.1/760
4	Main Status register (ADC_MSR)	32	R	0000_0001h	34.5.2/763
10	Interrupt Status Register (ADC_ISR)	32	R/W	0000_0000h	34.5.3/765
14	Channel Pending register 0 (ADC_CEOCFR0)	32	R/W	0000_0000h	34.5.4/766
18	Channel Pending register 1 (ADC_CEOCFR1)	32	R/W	0000_0000h	34.5.5/769
1C	Channel Pending register 2 (ADC_CEOCFR2)	32	R/W	0000_0000h	34.5.6/772
20	Interrupt Mask Register (ADC_IMR)	32	R/W	0000_0000h	34.5.7/775
24	Channel Interrupt Mask Register 0 (ADC_CIMR0)	32	R/W	0000_0000h	34.5.8/775
28	Channel Interrupt Mask Register 1 (ADC_CIMR1)	32	R/W	0000_0000h	34.5.9/778
2C	Channel Interrupt Mask Register 2 (ADC_CIMR2)	32	R/W	0000_0000h	34.5.10/ 780
30	Watchdog Threshold Interrupt Status Register (ADC_WTISR)	32	R/W	0000_0000h	34.5.11/ 783
34	Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)	32	R/W	0000_0000h	34.5.12/ 785
40	DMA Enable register (ADC_DMAE)	32	R/W	0000_0000h	34.5.13/ 786
44	DMA Channel Select Register 0 (ADC_DMAR0)	32	R/W	0000_0000h	34.5.14/ 787
48	DMA Channel Select Register 1 (ADC_DMAR1)	32	R/W	0000_0000h	34.5.15/ 789
4C	DMA Channel Select Register 2 (ADC_DMAR2)	32	R/W	0000_0000h	34.5.16/ 792
60	Threshold Register (ADC_THRHLR0)	32	R/W	0FFF_0000h	34.5.17/ 794
64	Threshold Register (ADC_THRHLR1)	32	R/W	0FFF_0000h	34.5.17/ 794
68	Threshold Register (ADC_THRHLR2)	32	R/W	0FFF_0000h	34.5.17/ 794
6C	Threshold Register (ADC_THRHLR3)	32	R/W	0FFF_0000h	34.5.17/ 794
80	Presampling Control Register (ADC_PSCR)	32	R/W	0000_0000h	34.5.18/ 795
84	Presampling register 0 (ADC_PSR0)	32	R/W	0000_0000h	34.5.19/ 796
88	Presampling register 1 (ADC_PSR1)	32	R/W	0000_0000h	34.5.20/ 798
8C	Presampling register 2 (ADC_PSR2)	32	R/W	0000_0000h	34.5.21/ 801
94	Conversion Timing Register 0 (ADC_CTR0)	32	R/W	0000_0014h	34.5.22/ 803
98	Conversion Timing Register 1 (ADC_CTR1)	32	R/W	0000_0014h	34.5.23/ 804

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9C	Conversion Timing Register 2 (ADC_CTR2)	32	R/W	0000_0016h	34.5.24/ 804
A4	Normal Conversion Mask Register 0 (ADC_NCMR0)	32	R/W	0000_0000h	34.5.25/ 805
A8	Normal Conversion Mask Register 1 (ADC_NCMR1)	32	R/W	0000_0000h	34.5.26/ 807
AC	Normal Conversion Mask Register 2 (ADC_NCMR2)	32	R/W	0000_0000h	34.5.27/ 810
B4	Injected Conversion Mask Register 0 (ADC_JCMR0)	32	R/W	0000_0000h	34.5.28/ 812
B8	Injected Conversion Mask Register 1 (ADC_JCMR1)	32	R/W	0000_0000h	34.5.29/ 814
BC	Injected Conversion Mask Register 2 (ADC_JCMR2)	32	R/W	0000_0000h	34.5.30/ 817
C4	Decode Signals Delay Register (ADC_DSDR)	32	R/W	0000_0000h	34.5.31/ 820
C8	Power Down Exit Delay Register (ADC_PDEDR)	32	R/W	0000_0000h	34.5.32/ 820
100	Channel Data Register n (Precision Channels) (ADC_CDR0)	32	R	0000_0000h	34.5.33/ 821
104	Channel Data Register n (Precision Channels) (ADC_CDR1)	32	R	0000_0000h	34.5.33/ 821
108	Channel Data Register n (Precision Channels) (ADC_CDR2)	32	R	0000_0000h	34.5.33/ 821
10C	Channel Data Register n (Precision Channels) (ADC_CDR3)	32	R	0000_0000h	34.5.33/ 821
110	Channel Data Register n (Precision Channels) (ADC_CDR4)	32	R	0000_0000h	34.5.33/ 821
114	Channel Data Register n (Precision Channels) (ADC_CDR5)	32	R	0000_0000h	34.5.33/ 821
118	Channel Data Register n (Precision Channels) (ADC_CDR6)	32	R	0000_0000h	34.5.33/ 821
11C	Channel Data Register n (Precision Channels) (ADC_CDR7)	32	R	0000_0000h	34.5.33/ 821
120	Channel Data Register n (Precision Channels) (ADC_CDR8)	32	R	0000_0000h	34.5.33/ 821
124	Channel Data Register n (Precision Channels) (ADC_CDR9)	32	R	0000_0000h	34.5.33/ 821
128	Channel Data Register n (Precision Channels) (ADC_CDR10)	32	R	0000_0000h	34.5.33/ 821
12C	Channel Data Register n (Precision Channels) (ADC_CDR11)	32	R	0000_0000h	34.5.33/ 821
130	Channel Data Register n (Precision Channels) (ADC_CDR12)	32	R	0000_0000h	34.5.33/ 821

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
134	Channel Data Register n (Precision Channels) (ADC_CDR13)	32	R	0000_0000h	34.5.33/ 821
138	Channel Data Register n (Precision Channels) (ADC_CDR14)	32	R	0000_0000h	34.5.33/ 821
13C	Channel Data Register n (Precision Channels) (ADC_CDR15)	32	R	0000_0000h	34.5.33/ 821
140	Channel Data Register n (Precision Channels) (ADC_CDR16)	32	R	0000_0000h	34.5.33/ 821
144	Channel Data Register n (Precision Channels) (ADC_CDR17)	32	R	0000_0000h	34.5.33/ 821
148	Channel Data Register n (Precision Channels) (ADC_CDR18)	32	R	0000_0000h	34.5.33/ 821
14C	Channel Data Register n (Precision Channels) (ADC_CDR19)	32	R	0000_0000h	34.5.33/ 821
150	Channel Data Register n (Precision Channels) (ADC_CDR20)	32	R	0000_0000h	34.5.33/ 821
154	Channel Data Register n (Precision Channels) (ADC_CDR21)	32	R	0000_0000h	34.5.33/ 821
158	Channel Data Register n (Precision Channels) (ADC_CDR22)	32	R	0000_0000h	34.5.33/ 821
15C	Channel Data Register n (Precision Channels) (ADC_CDR23)	32	R	0000_0000h	34.5.33/ 821
160	Channel Data Register n (Precision Channels) (ADC_CDR24)	32	R	0000_0000h	34.5.33/ 821
164	Channel Data Register n (Precision Channels) (ADC_CDR25)	32	R	0000_0000h	34.5.33/ 821
168	Channel Data Register n (Precision Channels) (ADC_CDR26)	32	R	0000_0000h	34.5.33/ 821
16C	Channel Data Register n (Precision Channels) (ADC_CDR27)	32	R	0000_0000h	34.5.33/ 821
170	Channel Data Register n (Precision Channels) (ADC_CDR28)	32	R	0000_0000h	34.5.33/ 821
174	Channel Data Register n (Precision Channels) (ADC_CDR29)	32	R	0000_0000h	34.5.33/ 821
178	Channel Data Register n (Precision Channels) (ADC_CDR30)	32	R	0000_0000h	34.5.33/ 821
17C	Channel Data Register n (Precision Channels) (ADC_CDR31)	32	R	0000_0000h	34.5.33/ 821
180	Channel Data Register n (Internal Channels) (ADC_CDR32)	32	R	0000_0000h	34.5.34/ 823
184	Channel Data Register n (Internal Channels) (ADC_CDR33)	32	R	0000_0000h	34.5.34/ 823
188	Channel Data Register n (Internal Channels) (ADC_CDR34)	32	R	0000_0000h	34.5.34/ 823

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
18C	Channel Data Register n (Internal Channels) (ADC_CDR35)	32	R	0000_0000h	34.5.34/ 823
190	Channel Data Register n (Internal Channels) (ADC_CDR36)	32	R	0000_0000h	34.5.34/ 823
194	Channel Data Register n (Internal Channels) (ADC_CDR37)	32	R	0000_0000h	34.5.34/ 823
198	Channel Data Register n (Internal Channels) (ADC_CDR38)	32	R	0000_0000h	34.5.34/ 823
19C	Channel Data Register n (Internal Channels) (ADC_CDR39)	32	R	0000_0000h	34.5.34/ 823
1A0	Channel Data Register n (Internal Channels) (ADC_CDR40)	32	R	0000_0000h	34.5.34/ 823
1A4	Channel Data Register n (Internal Channels) (ADC_CDR41)	32	R	0000_0000h	34.5.34/ 823
1A8	Channel Data Register n (Internal Channels) (ADC_CDR42)	32	R	0000_0000h	34.5.34/ 823
1AC	Channel Data Register n (Internal Channels) (ADC_CDR43)	32	R	0000_0000h	34.5.34/ 823
1B0	Channel Data Register n (Internal Channels) (ADC_CDR44)	32	R	0000_0000h	34.5.34/ 823
1B4	Channel Data Register n (Internal Channels) (ADC_CDR45)	32	R	0000_0000h	34.5.34/ 823
1B8	Channel Data Register n (Internal Channels) (ADC_CDR46)	32	R	0000_0000h	34.5.34/ 823
1BC	Channel Data Register n (Internal Channels) (ADC_CDR47)	32	R	0000_0000h	34.5.34/ 823
1C0	Channel Data Register n (Internal Channels) (ADC_CDR48)	32	R	0000_0000h	34.5.34/ 823
1C4	Channel Data Register n (Internal Channels) (ADC_CDR49)	32	R	0000_0000h	34.5.34/ 823
1C8	Channel Data Register n (Internal Channels) (ADC_CDR50)	32	R	0000_0000h	34.5.34/ 823
1CC	Channel Data Register n (Internal Channels) (ADC_CDR51)	32	R	0000_0000h	34.5.34/ 823
1D0	Channel Data Register n (Internal Channels) (ADC_CDR52)	32	R	0000_0000h	34.5.34/ 823
1D4	Channel Data Register n (Internal Channels) (ADC_CDR53)	32	R	0000_0000h	34.5.34/ 823
1D8	Channel Data Register n (Internal Channels) (ADC_CDR54)	32	R	0000_0000h	34.5.34/ 823
1DC	Channel Data Register n (Internal Channels) (ADC_CDR55)	32	R	0000_0000h	34.5.34/ 823
1E0	Channel Data Register n (Internal Channels) (ADC_CDR56)	32	R	0000_0000h	34.5.34/ 823

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1E4	Channel Data Register n (Internal Channels) (ADC_CDR57)	32	R	0000_0000h	34.5.34/ 823
1E8	Channel Data Register n (Internal Channels) (ADC_CDR58)	32	R	0000_0000h	34.5.34/ 823
1EC	Channel Data Register n (Internal Channels) (ADC_CDR59)	32	R	0000_0000h	34.5.34/ 823
1F0	Channel Data Register n (Internal Channels) (ADC_CDR60)	32	R	0000_0000h	34.5.34/ 823
1F4	Channel Data Register n (Internal Channels) (ADC_CDR61)	32	R	0000_0000h	34.5.34/ 823
1F8	Channel Data Register n (Internal Channels) (ADC_CDR62)	32	R	0000_0000h	34.5.34/ 823
1FC	Channel Data Register n (Internal Channels) (ADC_CDR63)	32	R	0000_0000h	34.5.34/ 823
200	Channel Data Register n (External Channels) (ADC_CDR64)	32	R	0000_0000h	34.5.35/ 825
204	Channel Data Register n (External Channels) (ADC_CDR65)	32	R	0000_0000h	34.5.35/ 825
208	Channel Data Register n (External Channels) (ADC_CDR66)	32	R	0000_0000h	34.5.35/ 825
20C	Channel Data Register n (External Channels) (ADC_CDR67)	32	R	0000_0000h	34.5.35/ 825
210	Channel Data Register n (External Channels) (ADC_CDR68)	32	R	0000_0000h	34.5.35/ 825
214	Channel Data Register n (External Channels) (ADC_CDR69)	32	R	0000_0000h	34.5.35/ 825
218	Channel Data Register n (External Channels) (ADC_CDR70)	32	R	0000_0000h	34.5.35/ 825
21C	Channel Data Register n (External Channels) (ADC_CDR71)	32	R	0000_0000h	34.5.35/ 825
220	Channel Data Register n (External Channels) (ADC_CDR72)	32	R	0000_0000h	34.5.35/ 825
224	Channel Data Register n (External Channels) (ADC_CDR73)	32	R	0000_0000h	34.5.35/ 825
228	Channel Data Register n (External Channels) (ADC_CDR74)	32	R	0000_0000h	34.5.35/ 825
22C	Channel Data Register n (External Channels) (ADC_CDR75)	32	R	0000_0000h	34.5.35/ 825
230	Channel Data Register n (External Channels) (ADC_CDR76)	32	R	0000_0000h	34.5.35/ 825
234	Channel Data Register n (External Channels) (ADC_CDR77)	32	R	0000_0000h	34.5.35/ 825
238	Channel Data Register n (External Channels) (ADC_CDR78)	32	R	0000_0000h	34.5.35/ 825

Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
23C	Channel Data Register n (External Channels) (ADC_CDR79)	32	R	0000_0000h	34.5.35/ 825
240	Channel Data Register n (External Channels) (ADC_CDR80)	32	R	0000_0000h	34.5.35/ 825
244	Channel Data Register n (External Channels) (ADC_CDR81)	32	R	0000_0000h	34.5.35/ 825
248	Channel Data Register n (External Channels) (ADC_CDR82)	32	R	0000_0000h	34.5.35/ 825
24C	Channel Data Register n (External Channels) (ADC_CDR83)	32	R	0000_0000h	34.5.35/ 825
250	Channel Data Register n (External Channels) (ADC_CDR84)	32	R	0000_0000h	34.5.35/ 825
254	Channel Data Register n (External Channels) (ADC_CDR85)	32	R	0000_0000h	34.5.35/ 825
258	Channel Data Register n (External Channels) (ADC_CDR86)	32	R	0000_0000h	34.5.35/ 825
25C	Channel Data Register n (External Channels) (ADC_CDR87)	32	R	0000_0000h	34.5.35/ 825
260	Channel Data Register n (External Channels) (ADC_CDR88)	32	R	0000_0000h	34.5.35/ 825
264	Channel Data Register n (External Channels) (ADC_CDR89)	32	R	0000_0000h	34.5.35/ 825
268	Channel Data Register n (External Channels) (ADC_CDR90)	32	R	0000_0000h	34.5.35/ 825
26C	Channel Data Register n (External Channels) (ADC_CDR91)	32	R	0000_0000h	34.5.35/ 825
270	Channel Data Register n (External Channels) (ADC_CDR92)	32	R	0000_0000h	34.5.35/ 825
274	Channel Data Register n (External Channels) (ADC_CDR93)	32	R	0000_0000h	34.5.35/ 825
278	Channel Data Register n (External Channels) (ADC_CDR94)	32	R	0000_0000h	34.5.35/ 825
27C	Channel Data Register n (External Channels) (ADC_CDR95)	32	R	0000_0000h	34.5.35/ 825
280	Threshold Register (ADC_THRHLR4)	32	R/W	0FFF_0000h	34.5.36/ 826
284	Threshold Register (ADC_THRHLR5)	32	R/W	0FFF_0000h	34.5.36/ 826
2B0	Channel Watchdog Select Register 0 (ADC_CWSELR0)	32	R/W	0000_0000h	34.5.37/ 826
2B4	Channel Watchdog Select Register 1 (ADC_CWSELR1)	32	R/W	0000_0000h	34.5.38/ 827
2B8	Channel Watchdog Select Register 2 (ADC_CWSELR2)	32	R/W	0000_0000h	34.5.39/ 828

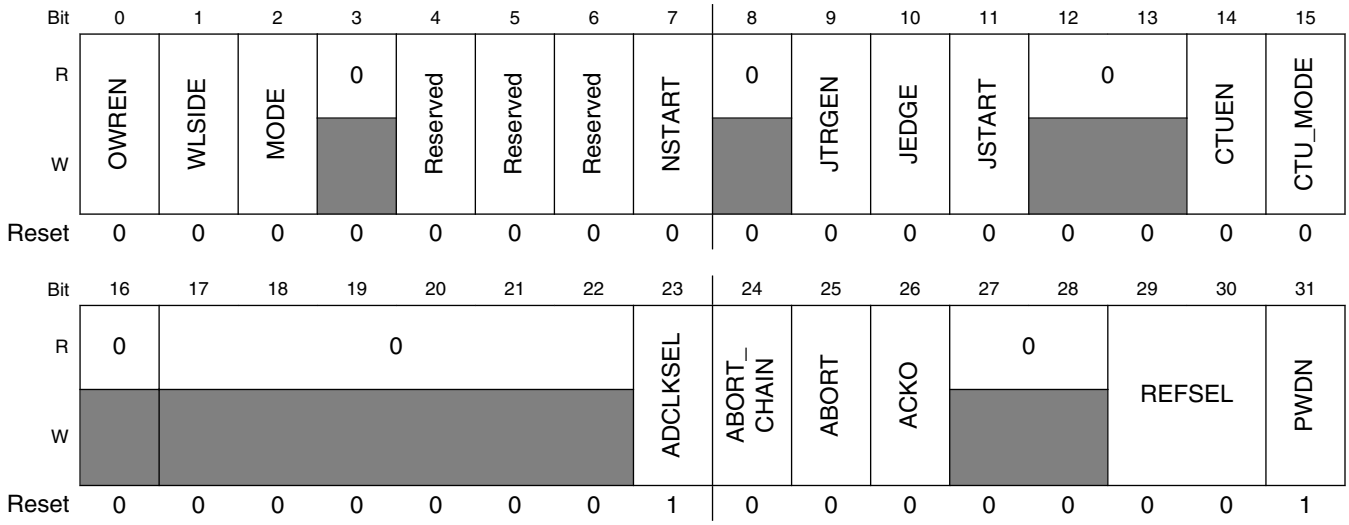
Table continues on the next page...

ADC memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2C0	Channel Watchdog Select Register 4 (ADC_CWSELR4)	32	R/W	0000_0000h	34.5.40/ 829
2C4	Channel Watchdog Select Register 5 (ADC_CWSELR5)	32	R/W	0000_0000h	34.5.41/ 830
2C8	Channel Watchdog Select Register 6 (ADC_CWSELR6)	32	R/W	0000_0000h	34.5.42/ 831
2CC	Channel Watchdog Select Register 7 (ADC_CWSELR7)	32	R/W	0000_0000h	34.5.43/ 832
2D0	Channel Watchdog Select Register 8 (ADC_CWSELR8)	32	R/W	0000_0000h	34.5.44/ 833
2D4	Channel Watchdog Select Register 9 (ADC_CWSELR9)	32	R/W	0000_0000h	34.5.45/ 834
2D8	Channel Watchdog Select Register 10 (ADC_CWSELR10)	32	R/W	0000_0000h	34.5.46/ 835
2DC	Channel Watchdog Select Register 11 (ADC_CWSELR11)	32	R/W	0000_0000h	34.5.47/ 836
2E0	Channel Watchdog Enable Register 0 (ADC_CWENR0)	32	R/W	0000_0000h	34.5.48/ 837
2E4	Channel Watchdog Enable Register 1 (ADC_CWENR1)	32	R/W	0000_0000h	34.5.49/ 840
2E8	Channel Watchdog Enable Register 2 (ADC_CWENR2)	32	R/W	0000_0000h	34.5.50/ 842
2F0	Analog Watchdog Out of Range Register 0 (ADC_AWORR0)	32	R/W	0000_0000h	34.5.51/ 845
2F4	Analog Watchdog Out of Range Register 1 (ADC_AWORR1)	32	R/W	0000_0000h	34.5.52/ 848
2F8	Analog Watchdog Out of Range Register 2 (ADC_AWORR2)	32	R/W	0000_0000h	34.5.53/ 850
3A0	Calibration, BIST Control and status Register (ADC_CALBISTREG)	32	R/W	C737_06F6h	34.5.54/ 853
3A8	Offset and Gain User Register (ADC_OFSGNUSR)	32	R/W	0000_0000h	34.5.55/ 855

34.5.1 Main Configuration Register (ADC_MCR)

Address: 0h base + 0h offset = 0h



ADC_MCR field descriptions

Field	Description
0 OWREN	Overwrite enable. 0 Older valid conversion data is not overwritten by newer conversion data. 1 Newer conversion result is always overwritten, irrespective of the validity of older conversion data.
1 WLSIDE	Write Left/Right aligned. 0 The conversion data is written right aligned from (32-resolution) to 31). 1 Data is left aligned (from 16 to (16 + resolution - 1).
2 MODE	One_Shot/Scan. 0 One_Shot Mode: configure the Normal conversion of one chain. 1 Scan Mode: configure continuous chain conversion mode; when the programmed chain conversion is finished it restarts immediately.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 Reserved	This field is reserved. Writing a value to this field has no effect.
5 Reserved	This field is reserved. Writing a value to this field has no effect.
6 Reserved	This field is reserved. **NOTE:** User must not overwrite default value of this field.
7 NSTART	Start of Normal conversion. Setting this field starts a normal conversion. In One shot mode: This field is cleared as soon as conversion is started. Setting this field during any ongoing conversion keeps this bit high until the requested conversion is started.

Table continues on the next page...

ADC_MCR field descriptions (continued)

Field	Description
	NOTE: This bit cannot be set (1) when PWDN = 1. This bit is always cleared when PWDN = 1.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 JTRGEN	Injection external trigger enable. When set, enables the external trigger to start a injected conversion; if the external trigger feature is needed for injected conversion then this field must be set.
10 JEDGE	Injection trigger edge selection. Edge selection for external trigger; if JTRGEN is one this bit selects the falling (JEDGE = zero) or rising (JEDGE = one) edge for the external trigger.
11 JSTART	Injection start. Setting this field will start the configured injected conversion chain. Resetting this field has no effect, as the injected chain conversion cannot be interrupted. Setting this field during ongoing Injected conversion keeps this field high value until the requested conversion is started.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CTUEN	Crosstrigger Unit Enable. Enables the crosstriggering unit (BCTU). 0 The crosstriggering unit is disabled and the BCTU-triggered injected conversion cannot take place. 1 The crosstriggering unit is enabled and the BCTUtriggered injected conversion can take place.
15 CTU_MODE	Crosstrigger Unit MODE bit When both modes are present in an ADC instance, this bit sets the operating mode of the ADC with respect to the BCTU. Please see the chip-specific ADC section for whether a specific ADC instance has both modes. Otherwise, treat this bit as Reserved. 0 BCTU control mode is selected. 1 BCTU trigger mode is selected.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 ADCLKSEL	Analog Clock frequency Selector. This field can be written in Power Down mode only. Please check device specific additional restriction on programming of this field. 0 ADC clock frequency is half of bus clock 1 ADC clock frequency is equal to bus clock
24 ABORT_CHAIN	Abort Chain. If this bit is set the ongoing Chain Conversion is aborted. This bit gets reset by hardware as soon as a new conversion is requested. **NOTE:** Setting this bit in idle state (no normal/injected conversion is ongoing) has no effect. In this case it is reset immediately. During ongoing BCTU conversion this bit also cannot be programmed.

Table continues on the next page...

ADC_MCR field descriptions (continued)

Field	Description
25 ABORT	Abort Conversion. If this bit is set, the ongoing channel conversion is aborted and the next channel conversion is invoked. This bit gets reset by hardware as soon as the new conversion is invoked. **NOTE:** Setting this bit in idle state (No Normal/injected conversion is ongoing) has no effect. In this case it is reset immediately. During ongoing BCTU conversion this bit cannot be programmed
26 ACKO	Auto clock off enable. If set, enables the Auto clock off feature.
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 REFSEL	Reference voltage selection for ADC analog part. 00 Select VREFH as reference voltage [1] 01 Reserved 10 Reserved 11 Reserved
31 PWDN	Power-down enable. When this bit is set, the analog module is requested to enter Power-Down mode. When the ADC's status is PWDN, resetting this bit will start the ADC's transition to IDLE mode. In BCTU control mode MCR[CTUEN] must be reset before entering into power-down mode.

1. For ADC0, VREFH corresponds to VDD_HV_ADC0
   For ADC1, VREFH corresponds to VDD_HV_ADC1_REF

34.5.2 Main Status register (ADC_MSR)

Address: 0h base + 4h offset = 4h



ADC_MSR field descriptions

Field	Description
0 CALIBRTD	This bit indicates the ADC calibration status. This bit gets updated after running the High Accuracy Calibration mode. 0 Uncalibrated or calibration unsuccessful 1 Calibration was run and it was successful

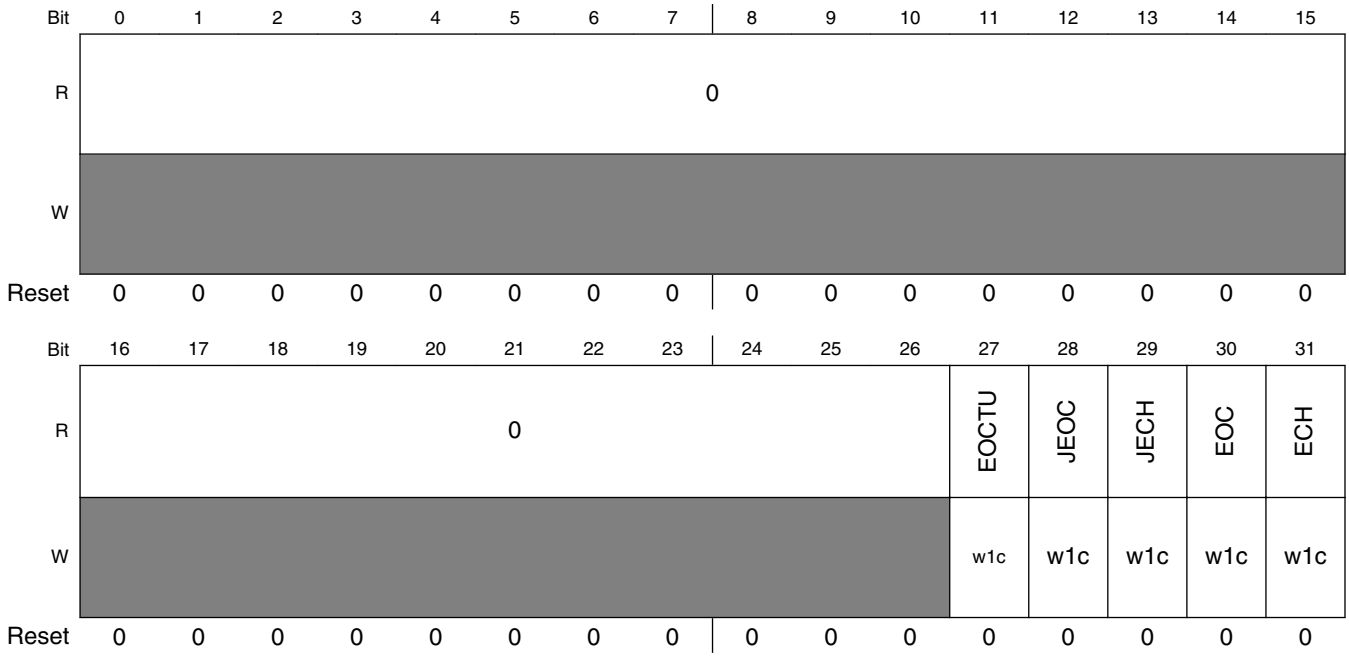
Table continues on the next page...

ADC_MSR field descriptions (continued)

Field	Description
1–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 NSTART	This status bit is used to signal that a Normal conversion is ongoing.
8 JABORT	This status bit is used to signal that an Injected conversion has been aborted by BCTU. That bit is reset when a new injected conversion starts.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 JSTART	This status bit is used to signal that an Injected conversion is ongoing.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 CTUSTART	This status bit is used to signal that a BCTU conversion is ongoing. This bit is set when a BCTU trigger pulse is received and the BCTU conversion starts.When BCTU trigger mode is enabled this bit is automatically reset when the conversion is completed.When Control Mode is enabled this bit is reset when the BCTU is disabled (CTUEN set to 0).
16–22 CHADDR	Channel under measure address. Status bits are used to signal which channel is under measure. MSR[CHADDR] is only valid when the SARADC is in the sample phase of the conversion (MSR[ADCSTATUS] = 100b).
23–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 ACKO	Auto clock off enable. This status bit is used to signal if the Auto clock off feature is on.
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 ADCSTATUS	Status of the ADC. 000 Idle 001 Power Down 100 Sample 110 Conversion 010 Wait state (waiting to start conversion [external trigger]) The start of ADC conversion can be delayed (after a programmed duration in "DSDR" register) for a time after the application of trigger for External channels. During this wait period the ADCSTATUS is shown as "wait state". 011 Busy in Calibration

34.5.3 Interrupt Status Register (ADC_ISR)

Address: 0h base + 10h offset = 10h



ADC_ISR field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 EOCTU	End of BCTU conversion. This bit is set for a digital end of conversion on a the BCTU Channel; active when set.
28 JEOC	End of injected channel conversion. This bit is set for a digital end of conversion on an injected channel; active when set.
29 JECH	End of injected chain conversion. This bit is set for a digital end of chain conversion on an injected channel; active when set. **NOTE:** If there is no channel selected in the ADC_JCMR*n* register(s) and there is a start of conversion trigger, then ADC_ISR[JECH] is set and a JECH interrupt is immediately issued (if enabled). Similarly, if no channel is selected in the ADC_NCMR*n* register and a normal conversion is triggered, ADC_ISR[ECH] is set and an ECH interrupt is immediately issued, if enabled.
30 EOC	End of channel conversion. This bit is set for a digital end of conversion on a Normal Channel; active when set.
31 ECH	End of chain conversion. This bit is set for a digital end of chain conversion on a Normal Channel; active when set. **NOTE:** If there is no channel selected in the ADC_NCMR*n* register(s) and there is a start of conversion trigger, then ADC_ISR[ECH] is set and an ECH interrupt is immediately issued (if enabled).

Table continues on the next page...

ADC_ISR field descriptions (continued)

Field	Description
	Similarly, if no channel is selected in the ADC_JCMR*n* register and an injected conversion is triggered, ADC_ISR[JECH] is set and a JECH interrupt is immediately issued, if enabled.

34.5.4 Channel Pending register 0 (ADC_CEOCFR0)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOCCH31	EOCCH30	EOCCH29	EOCCH28	EOCCH27	EOCCH26	EOCCH25	EOCCH24	EOCCH23	EOCCH22	EOCCH21	EOCCH20	EOCCH19	EOCCH18	EOCCH17	EOCCH16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOCCH15	EOCCH14	EOCCH13	EOCCH12	EOCCH11	EOCCH10	EOCCH9	EOCCH8	EOCCH7	EOCCH6	EOCCH5	EOCCH4	EOCCH3	EOCCH2	EOCCH1	EOCCH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CEOCFR0 field descriptions

Field	Description
0 EOCCH31	EOC Channel 31. When set, the measure of channel 31 is completed.
1 EOCCH30	EOC Channel 30. When set, the measure of channel 30 is completed.

Table continues on the next page...

ADC_CEOCFR0 field descriptions (continued)

Field	Description
2 EOCCH29	EOC Channel 29. When set, the measure of channel 29 is completed.
3 EOCCH28	EOC Channel 28. When set, the measure of channel 28 is completed.
4 EOCCH27	EOC Channel 27. When set, the measure of channel 27 is completed.
5 EOCCH26	EOC Channel 26. When set, the measure of channel 26 is completed.
6 EOCCH25	EOC Channel 25. When set, the measure of channel 25 is completed.
7 EOCCH24	EOC Channel 24. When set, the measure of channel 24 is completed.
8 EOCCH23	EOC Channel 23. When set, the measure of channel 23 is completed.
9 EOCCH22	EOC Channel 22. When set, the measure of channel 22 is completed.
10 EOCCH21	EOC Channel 21. When set, the measure of channel 21 is completed.
11 EOCCH20	EOC Channel 20. When set, the measure of channel 20 is completed.
12 EOCCH19	EOC Channel 19. When set, the measure of channel 19 is completed.
13 EOCCH18	EOC Channel 18. When set, the measure of channel 18 is completed.
14 EOCCH17	EOC Channel 17. When set, the measure of channel 17 is completed.
15 EOCCH16	EOC Channel 16. When set, the measure of channel 16 is completed.
16 EOCCH15	EOC Channel 15. When set, the measure of channel 15 is completed.
17 EOCCH14	EOC Channel 14. When set, the measure of channel 14 is completed.
18 EOCCH13	EOC Channel 13. When set, the measure of channel 13 is completed.
19 EOCCH12	EOC Channel 12.

Table continues on the next page...

ADC_CEOCFR0 field descriptions (continued)

Field	Description
	When set, the measure of channel 12 is completed.
20 EOCCH11	EOC Channel 11. When set, the measure of channel 11 is completed.
21 EOCCH10	EOC Channel 10. When set, the measure of channel 10 is completed.
22 EOCCH9	EOC Channel 9. When set, the measure of channel 9 is completed.
23 EOCCH8	EOC Channel 8. When set, the measure of channel 8 is completed.
24 EOCCH7	EOC Channel 7. When set, the measure of channel 7 is completed.
25 EOCCH6	EOC Channel 6. When set, the measure of channel 6 is completed.
26 EOCCH5	EOC Channel 5. When set, the measure of channel 5 is completed.
27 EOCCH4	EOC Channel 4. When set, the measure of channel 4 is completed.
28 EOCCH3	EOC Channel 3. When set, the measure of channel 3 is completed.
29 EOCCH2	EOC Channel 2. When set, the measure of channel 2 is completed.
30 EOCCH1	EOC Channel 1. When set, the measure of channel 1 is completed.
31 EOCCH0	EOC Channel 0. When set, the measure of channel 0 is completed.

34.5.5 Channel Pending register 1 (ADC_CEOCFR1)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOCCH63	EOCCH62	EOCCH61	EOCCH60	EOCCH59	EOCCH58	EOCCH57	EOCCH56	EOCCH55	EOCCH54	EOCCH53	EOCCH52	EOCCH51	EOCCH50	EOCCH49	EOCCH48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOCCH47	EOCCH46	EOCCH45	EOCCH44	EOCCH43	EOCCH42	EOCCH41	EOCCH40	EOCCH39	EOCCH38	EOCCH37	EOCCH36	EOCCH35	EOCCH34	EOCCH33	EOCCH32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CEOCFR1 field descriptions

Field	Description
0 EOCCH63	EOC Channel 63. When set, the measure of channel 63 is completed.
1 EOCCH62	EOC Channel 62. When set, the measure of channel 62 is completed.
2 EOCCH61	EOC Channel 61. When set, the measure of channel 61 is completed.
3 EOCCH60	EOC Channel 60. When set, the measure of channel 60 is completed.

Table continues on the next page...

ADC_CEOCFR1 field descriptions (continued)

Field	Description
4 EOCCH59	EOC Channel 59. When set, the measure of channel 59 is completed.
5 EOCCH58	EOC Channel 58. When set, the measure of channel 58 is completed.
6 EOCCH57	EOC Channel 57. When set, the measure of channel 57 is completed.
7 EOCCH56	EOC Channel 56. When set, the measure of channel 56 is completed.
8 EOCCH55	EOC Channel 55. When set, the measure of channel 55 is completed.
9 EOCCH54	EOC Channel 54. When set, the measure of channel 54 is completed.
10 EOCCH53	EOC Channel 53. When set, the measure of channel 53 is completed.
11 EOCCH52	EOC Channel 52. When set, the measure of channel 52 is completed.
12 EOCCH51	EOC Channel 51. When set, the measure of channel 51 is completed.
13 EOCCH50	EOC Channel 50. When set, the measure of channel 50 is completed.
14 EOCCH49	EOC Channel 49. When set, the measure of channel 49 is completed.
15 EOCCH48	EOC Channel 48 When set, the measure of channel 48 is completed.
16 EOCCH47	EOC Channel 47. When set, the measure of channel 47 is completed.
17 EOCCH46	EOC Channel 46. When set, the measure of channel 46 is completed.
18 EOCCH45	EOC Channel 45. When set, the measure of channel 45 is completed.
19 EOCCH44	EOC Channel 44. When set, the measure of channel 44 is completed.
20 EOCCH43	EOC Channel 43. When set, the measure of channel 43 is completed.
21 EOCCH42	EOC Channel 42.

Table continues on the next page...

ADC_CEOCFR1 field descriptions (continued)

Field	Description
	When set, the measure of channel 42 is completed.
22 EOCCH41	EOC Channel 41. When set, the measure of channel 41 is completed.
23 EOCCH40	EOC Channel 40. When set, the measure of channel 40 is completed.
24 EOCCH39	EOC Channel 39. When set, the measure of channel 39 is completed.
25 EOCCH38	EOC Channel 38. When set, the measure of channel 38 is completed.
26 EOCCH37	EOC Channel 37. When set, the measure of channel 37 is completed.
27 EOCCH36	EOC Channel 36. When set, the measure of channel 36 is completed.
28 EOCCH35	EOC Channel 35. When set, the measure of channel 35 is completed.
29 EOCCH34	EOC Channel 34. When set, the measure of channel 34 is completed.
30 EOCCH33	EOC Channel 33. When set, the measure of channel 33 is completed.
31 EOCCH32	EOC Channel 32. When set, the measure of channel 32 is completed.

34.5.6 Channel Pending register 2 (ADC_CEOCFR2)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOCCH95	EOCCH94	EOCCH93	EOCCH92	EOCCH91	EOCCH90	EOCCH89	EOCCH88	EOCCH87	EOCCH86	EOCCH85	EOCCH84	EOCCH83	EOCCH82	EOCCH81	EOCCH80
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EOCCH79	EOCCH78	EOCCH77	EOCCH76	EOCCH75	EOCCH74	EOCCH73	EOCCH72	EOCCH71	EOCCH70	EOCCH69	EOCCH68	EOCCH67	EOCCH66	EOCCH65	EOCCH64
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CEOCFR2 field descriptions

Field	Description
0 EOCCH95	EOC Channel 95. When set, the measure of channel 95 is completed.
1 EOCCH94	EOC Channel 94. When set, the measure of channel 94 is completed.
2 EOCCH93	EOC Channel 93. When set, the measure of channel 93 is completed.
3 EOCCH92	EOC Channel 92. When set, the measure of channel 92 is completed.

Table continues on the next page...

ADC_CEOCFR2 field descriptions (continued)

Field	Description
4 EOCCH91	EOC Channel 91. When set, the measure of channel 91 is completed.
5 EOCCH90	EOC Channel 90. When set, the measure of channel 90 is completed.
6 EOCCH89	EOC Channel 89. When set, the measure of channel 89 is completed.
7 EOCCH88	EOC Channel 88. When set, the measure of channel 88 is completed.
8 EOCCH87	EOC Channel 87. When set, the measure of channel 87 is completed.
9 EOCCH86	EOC Channel 86. When set, the measure of channel 86 is completed.
10 EOCCH85	EOC Channel 85. When set, the measure of channel 85 is completed.
11 EOCCH84	EOC Channel 84. When set, the measure of channel 84 is completed.
12 EOCCH83	EOC Channel 83. When set, the measure of channel 83 is completed.
13 EOCCH82	EOC Channel 82. When set, the measure of channel 82 is completed.
14 EOCCH81	EOC Channel 81. When set, the measure of channel 81 is completed.
15 EOCCH80	EOC Channel 80. When set, the measure of channel 80 is completed.
16 EOCCH79	EOC Channel 79. When set, the measure of channel 79 is completed.
17 EOCCH78	EOC Channel 78. When set, the measure of channel 78 is completed.
18 EOCCH77	EOC Channel 77. When set, the measure of channel 77 is completed.
19 EOCCH76	EOC Channel 76. When set, the measure of channel 76 is completed.
20 EOCCH75	EOC Channel 75. When set, the measure of channel 75 is completed.
21 EOCCH74	EOC Channel 74.

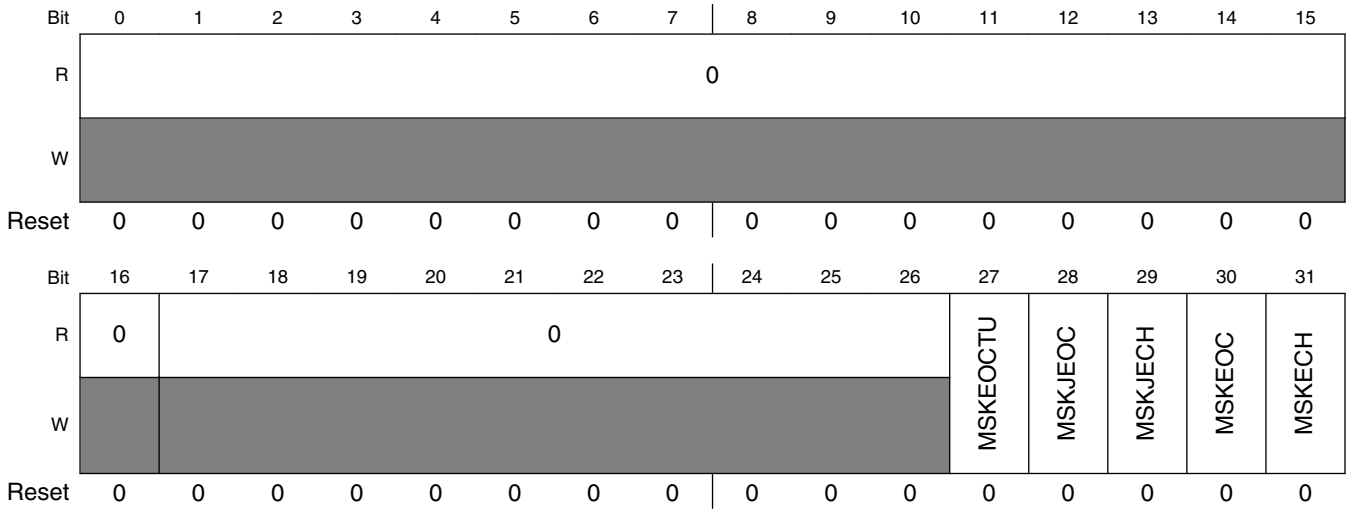
Table continues on the next page...

ADC_CEOCFR2 field descriptions (continued)

Field	Description
	When set, the measure of channel 74 is completed.
22 EOCCH73	EOC Channel 73. When set, the measure of channel 73 is completed.
23 EOCCH72	EOC Channel 72. When set, the measure of channel 72 is completed.
24 EOCCH71	EOC Channel 71. When set, the measure of channel 71 is completed.
25 EOCCH70	EOC Channel 70. When set, the measure of channel 70 is completed.
26 EOCCH69	EOC Channel 69. When set, the measure of channel 69 is completed.
27 EOCCH68	EOC Channel 68. When set, the measure of channel 68 is completed.
28 EOCCH67	EOC Channel 67. When set, the measure of channel 67 is completed.
29 EOCCH66	EOC Channel 66. When set, the measure of channel 66 is completed.
30 EOCCH65	EOC Channel 65. When set, the measure of channel 65 is completed.
31 EOCCH64	EOC Channel 64. When set, the measure of channel 64 is completed.

34.5.7 Interrupt Mask Register (ADC_IMR)

Address: 0h base + 20h offset = 20h



ADC_IMR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 MSKEOCTU	Mask bit for EOCTU. When set, the interrupt is enabled.
28 MSKJEOC	Mask bit for JEOC. When set, the interrupt is enabled.
29 MSKJECH	Mask bit for JECH. When set, the interrupt is enabled.
30 MSKEOC	Mask bit for EOC. When set, the interrupt is enabled.
31 MSKECH	Mask bit for ECH. When set, the interrupt is enabled.

34.5.8 Channel Interrupt Mask Register 0 (ADC_CIMR0)

The fields in this register globally enable or disable end of conversion interrupts by channels within the range of Channel 0 through 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific details to find which channels are applicable.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CIM31	CIM30	CIM29	CIM28	CIM27	CIM26	CIM25	CIM24	CIM23	CIM22	CIM21	CIM20	CIM19	CIM18	CIM17	CIM16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIM15	CIM14	CIM13	CIM12	CIM11	CIM10	CIM9	CIM8	CIM7	CIM6	CIM5	CIM4	CIM3	CIM2	CIM1	CIM0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CIMR0 field descriptions

Field	Description
0 CIM31	Channel 31 interrupt enable. When this field is set, the interrupt for channel 31 is enabled.
1 CIM30	Channel 30 interrupt enable. When this field is set, the interrupt for channel 30 is enabled.
2 CIM29	Channel 29 interrupt enable. When this field is set, the interrupt for channel 29 is enabled.
3 CIM28	Channel 28 interrupt enable. When this field is set, the interrupt for channel 28 is enabled.
4 CIM27	Channel 27 interrupt enable. When this field is set, the interrupt for channel 27 is enabled.
5 CIM26	Channel 26 interrupt enable. When this field is set, the interrupt for channel 26 is enabled.
6 CIM25	Channel 25 interrupt enable. When this field is set, the interrupt for channel 25 is enabled.
7 CIM24	Channel 24 interrupt enable. When this field is set, the interrupt for channel 24 is enabled.
8 CIM23	Channel 23 interrupt enable. When this field is set, the interrupt for channel 23 is enabled.

Table continues on the next page...

ADC_CIMR0 field descriptions (continued)

Field	Description
9 CIM22	Channel 22 interrupt enable. When this field is set, the interrupt for channel 22 is enabled.
10 CIM21	Channel 21 interrupt enable. When this field is set, the interrupt for channel 21 is enabled.
11 CIM20	Channel 20 interrupt enable. When this field is set, the interrupt for channel 20 is enabled.
12 CIM19	Channel 19 interrupt enable. When this field is set, the interrupt for channel 19 is enabled.
13 CIM18	Channel 18 interrupt enable. When this field is set, the interrupt for channel 18 is enabled.
14 CIM17	Channel 17 interrupt enable. When this field is set, the interrupt for channel 17 is enabled.
15 CIM16	Channel 16 interrupt enable. When this field is set, the interrupt for channel 16 is enabled.
16 CIM15	Channel 15 interrupt enable. When this field is set, the interrupt for channel 15 is enabled.
17 CIM14	Channel 14 interrupt enable. When this field is set, the interrupt for channel 14 is enabled.
18 CIM13	Channel 13 interrupt enable. When this field is set, the interrupt for channel 13 is enabled.
19 CIM12	Channel 12 interrupt enable. When this field is set, the interrupt for channel 12 is enabled.
20 CIM11	Channel 11 interrupt enable. When this field is set, the interrupt for channel 11 is enabled.
21 CIM10	Channel 10 interrupt enable. When this field is set, the interrupt for channel 10 is enabled.
22 CIM9	Channel 9 interrupt enable. When this field is set, the interrupt for channel 9 is enabled.
23 CIM8	Channel 8 interrupt enable. When this field is set, the interrupt for channel 8 is enabled.
24 CIM7	Channel 7 interrupt enable. When this field is set, the interrupt for channel 7 is enabled.
25 CIM6	Channel 6 interrupt enable. When this field is set, the interrupt for channel 6 is enabled.
26 CIM5	Channel 5 interrupt enable.

Table continues on the next page...

ADC_CIMR0 field descriptions (continued)

Field	Description
	When this field is set, the interrupt for channel 5 is enabled.
27 CIM4	Channel 4 interrupt enable. When this field is set, the interrupt for channel 4 is enabled.
28 CIM3	Channel 3 interrupt enable. When this field is set, the interrupt for channel 3 is enabled.
29 CIM2	Channel 2 interrupt enable. When this field is set, the interrupt for channel 2 is enabled.
30 CIM1	Channel 1 interrupt enable. When this field is set, the interrupt for channel 1 is enabled.
31 CIM0	Channel 0 interrupt enable. When this field is set, the interrupt for channel 0 is enabled.

34.5.9 Channel Interrupt Mask Register 1 (ADC_CIMR1)

The fields in this register globally enable or disable end of conversion interrupts by channels within the range of Channel 32 through 63.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific details to find which channels are applicable.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CIM63	CIM62	CIM61	CIM60	CIM59	CIM58	CIM57	CIM56	CIM55	CIM54	CIM53	CIM52	CIM51	CIM50	CIM49	CIM48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIM47	CIM46	CIM45	CIM44	CIM43	CIM42	CIM41	CIM40	CIM39	CIM38	CIM37	CIM36	CIM35	CIM34	CIM33	CIM32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CIMR1 field descriptions

Field	Description
0 CIM63	Channel 63 interrupt enable. When this field is set, the interrupt for channel 63 is enabled.
1 CIM62	Channel 62 interrupt enable. When this field is set, the interrupt for channel 62 is enabled.
2 CIM61	Channel 61 interrupt enable. When this field is set, the interrupt for channel 61 is enabled.
3 CIM60	Channel 60 interrupt enable. When this field is set, the interrupt for channel 60 is enabled.
4 CIM59	Channel 59 interrupt enable. When this field is set, the interrupt for channel 59 is enabled.
5 CIM58	Channel 58 interrupt enable. When this field is set, the interrupt for channel 58 is enabled.
6 CIM57	Channel 57 interrupt enable. When this field is set, the interrupt for channel 57 is enabled.
7 CIM56	Channel 56 interrupt enable. When this field is set, the interrupt for channel 56 is enabled.
8 CIM55	Channel 55 interrupt enable. When this field is set, the interrupt for channel 55 is enabled.
9 CIM54	Channel 54 interrupt enable. When this field is set, the interrupt for channel 54 is enabled.
10 CIM53	Channel 53 interrupt enable. When this field is set, the interrupt for channel 53 is enabled.
11 CIM52	Channel 52 interrupt enable. When this field is set, the interrupt for channel 52 is enabled.
12 CIM51	Channel 51 interrupt enable. When this field is set, the interrupt for channel 51 is enabled.
13 CIM50	Channel 50 interrupt enable. When this field is set, the interrupt for channel 50 is enabled.
14 CIM49	Channel 49 interrupt enable. When this field is set, the interrupt for channel 49 is enabled.
15 CIM48	Channel 48 interrupt enable. When this field is set, the interrupt for channel 48 is enabled.
16 CIM47	Channel 47 interrupt enable. When this field is set, the interrupt for channel 47 is enabled.
17 CIM46	Channel 46 interrupt enable.

Table continues on the next page...

ADC_CIMR1 field descriptions (continued)

Field	Description
	When this field is set, the interrupt for channel 46 is enabled.
18 CIM45	Channel 45 interrupt enable. When this field is set, the interrupt for channel 45 is enabled.
19 CIM44	Channel 44 interrupt enable. When this field is set, the interrupt for channel 44 is enabled.
20 CIM43	Channel 43 interrupt enable. When this field is set, the interrupt for channel 43 is enabled.
21 CIM42	Channel 42 interrupt enable. When this field is set, the interrupt for channel 42 is enabled.
22 CIM41	Channel 41 interrupt enable. When this field is set, the interrupt for channel 41 is enabled.
23 CIM40	Channel 40 interrupt enable. When this field is set, the interrupt for channel 40 is enabled.
24 CIM39	Channel 39 interrupt enable. When this field is set, the interrupt for channel 39 is enabled.
25 CIM38	Channel 38 interrupt enable. When this field is set, the interrupt for channel 38 is enabled.
26 CIM37	Channel 37 interrupt enable. When this field is set, the interrupt for channel 37 is enabled.
27 CIM36	Channel 36 interrupt enable. When this field is set, the interrupt for channel 36 is enabled.
28 CIM35	Channel 35 interrupt enable. When this field is set, the interrupt for channel 35 is enabled.
29 CIM34	Channel 34 interrupt enable. When this field is set, the interrupt for channel 34 is enabled.
30 CIM33	Channel 33 interrupt enable. When this field is set, the interrupt for channel 33 is enabled.
31 CIM32	Channel 32 interrupt enable. When this field is set, the interrupt for channel 32 is enabled.

34.5.10 Channel Interrupt Mask Register 2 (ADC_CIMR2)

The fields in this register globally enable or disable end of conversion interrupts by channels within the range of Channel 64 through 95.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific details to find which channels are applicable.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CIM95	CIM94	CIM93	CIM92	CIM91	CIM90	CIM89	CIM88	CIM87	CIM86	CIM85	CIM84	CIM83	CIM82	CIM81	CIM80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CIM79	CIM78	CIM77	CIM76	CIM75	CIM74	CIM73	CIM72	CIM71	CIM70	CIM69	CIM68	CIM67	CIM66	CIM65	CIM64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CIMR2 field descriptions

Field	Description
0 CIM95	Channel 95 interrupt enable. When this field is set, the interrupt for channel 95 is enabled.
1 CIM94	Channel 94 interrupt enable. When this field is set, the interrupt for channel 94 is enabled.
2 CIM93	Channel 93 interrupt enable. When this field is set, the interrupt for channel 93 is enabled.
3 CIM92	Channel 92 interrupt enable. When this field is set, the interrupt for channel 92 is enabled.
4 CIM91	Channel 91 interrupt enable. When this field is set, the interrupt for channel 91 is enabled.
5 CIM90	Channel 90 interrupt enable. When this field is set, the interrupt for channel 90 is enabled.
6 CIM89	Channel 89 interrupt enable. When this field is set, the interrupt for channel 89 is enabled.
7 CIM88	Channel 88 interrupt enable. When this field is set, the interrupt for channel 88 is enabled.
8 CIM87	Channel 87 interrupt enable. When this field is set, the interrupt for channel 87 is enabled.

Table continues on the next page...

ADC_CIMR2 field descriptions (continued)

Field	Description
9 CIM86	Channel 86 interrupt enable. When this field is set, the interrupt for channel 86 is enabled.
10 CIM85	Channel 85 interrupt enable. When this field is set, the interrupt for channel 85 is enabled.
11 CIM84	Channel 84 interrupt enable. When this field is set, the interrupt for channel 84 is enabled.
12 CIM83	Channel 83 interrupt enable. When this field is set, the interrupt for channel 83 is enabled.
13 CIM82	Channel 82 interrupt enable. When this field is set, the interrupt for channel 82 is enabled.
14 CIM81	Channel 81 interrupt enable. When this field is set, the interrupt for channel 81 is enabled.
15 CIM80	Channel 80 interrupt enable. When this field is set, the interrupt for channel 80 is enabled.
16 CIM79	Channel 79 interrupt enable. When this field is set, the interrupt for channel 79 is enabled.
17 CIM78	Channel 78 interrupt enable. When this field is set, the interrupt for channel 78 is enabled.
18 CIM77	Channel 77 interrupt enable. When this field is set, the interrupt for channel 77 is enabled.
19 CIM76	Channel 76 interrupt enable. When this field is set, the interrupt for channel 76 is enabled.
20 CIM75	Channel 75 interrupt enable. When this field is set, the interrupt for channel 75 is enabled.
21 CIM74	Channel 74 interrupt enable. When this field is set, the interrupt for channel 74 is enabled.
22 CIM73	Channel 73 interrupt enable. When this field is set, the interrupt for channel 73 is enabled.
23 CIM72	Channel 72 interrupt enable. When this field is set, the interrupt for channel 72 is enabled.
24 CIM71	Channel 71 interrupt enable. When this field is set, the interrupt for channel 71 is enabled.
25 CIM70	Channel 70 interrupt enable. When this field is set, the interrupt for channel 70 is enabled.
26 CIM69	Channel 69 interrupt enable.

Table continues on the next page...

ADC_CIMR2 field descriptions (continued)

Field	Description
	When this field is set, the interrupt for channel 69 is enabled.
27 CIM68	Channel 68 interrupt enable. When this field is set, the interrupt for channel 68 is enabled.
28 CIM67	Channel 67 interrupt enable. When this field is set, the interrupt for channel 67 is enabled.
29 CIM66	Channel 66 interrupt enable. When this field is set, the interrupt for channel 66 is enabled.
30 CIM65	Channel 65 interrupt enable. When this field is set, the interrupt for channel 65 is enabled.
31 CIM64	Channel 64 interrupt enable. When this field is set, the interrupt for channel 64 is enabled.

34.5.11 Watchdog Threshold Interrupt Status Register (ADC_WTISR)

The fields in this register, when set, indicate either the upper or lower threshold value has been crossed for a specific watchdog.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

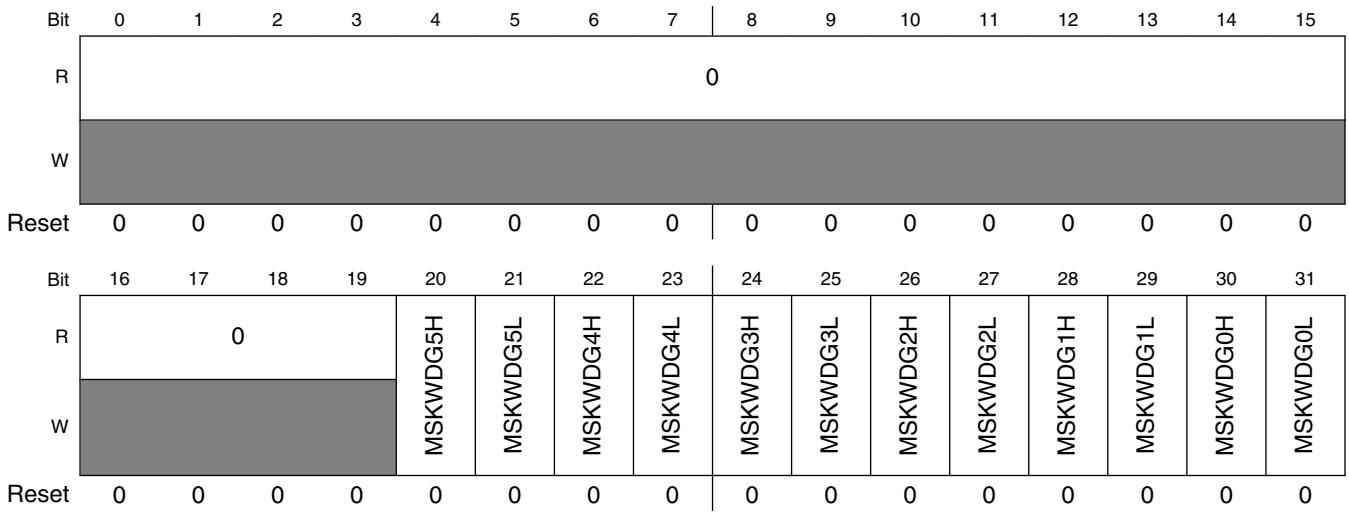
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				WDG5H	WDG5L	WDG4H	WDG4L	WDG3H	WDG3L	WDG2H	WDG2L	WDG1H	WDG1L	WDG0H	WDG0L
W					w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_WTISR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 WDG5H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 5 (WD5). **NOTE:** Write a '1' to this field to clear it.
21 WDG5L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 5 (WD5). **NOTE:** Write a '1' to this field to clear it.
22 WDG4H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 4 (WD4). **NOTE:** Write a '1' to this field to clear it.
23 WDG4L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 4 (WD4). **NOTE:** Write a '1' to this field to clear it.
24 WDG3H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 3 (WD3). **NOTE:** Write a '1' to this field to clear it.
25 WDG3L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 3 (WD3). **NOTE:** Write a '1' to this field to clear it.
26 WDG2H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 2 (WD2). **NOTE:** Write a '1' to this field to clear it.
27 WDG2L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 2 (WD2). **NOTE:** Write a '1' to this field to clear it.
28 WDG1H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 1 (WD1). **NOTE:** Write a '1' to this field to clear it.
29 WDG1L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 1 (WD1). **NOTE:** Write a '1' to this field to clear it.
30 WDG0H	This bit corresponds to the interrupt generated on the converted value being higher than the programmed higher threshold for a channel monitored by Watchdog 0 (WD0). **NOTE:** Write a '1' to this field to clear it.
31 WDG0L	This bit corresponds to the interrupt generated on the converted value being lower than the programmed lower threshold for a channel monitored by Watchdog 0 (WD0). **NOTE:** Write a '1' to this field to clear it.

34.5.12 Watchdog Threshold Interrupt Mask Register (ADC_WTIMR)

Address: 0h base + 34h offset = 34h



ADC_WTIMR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 MSKWDG5H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 5 (WD5) being higher than the programmed higher threshold. When set, the interrupt is enabled.
21 MSKWDG5L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 5 (WD5) being lower than the programmed lower threshold. When set, the interrupt is enabled.
22 MSKWDG4H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 4 (WD4) being higher than the programmed higher threshold. When set, the interrupt is enabled.
23 MSKWDG4L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 4 (WD4) being lower than the programmed lower threshold. When set, the interrupt is enabled.
24 MSKWDG3H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 3 (WD3) being higher than the programmed higher threshold. When set, the interrupt is enabled.
25 MSKWDG3L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 3 (WD3) being lower than the programmed lower threshold. When set, the interrupt is enabled.
26 MSKWDG2H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 2 (WD2) being higher than the programmed higher threshold. When set, the interrupt is enabled.
27 MSKWDG2L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 2 (WD2) being lower than the programmed lower threshold. When set, the interrupt is enabled.

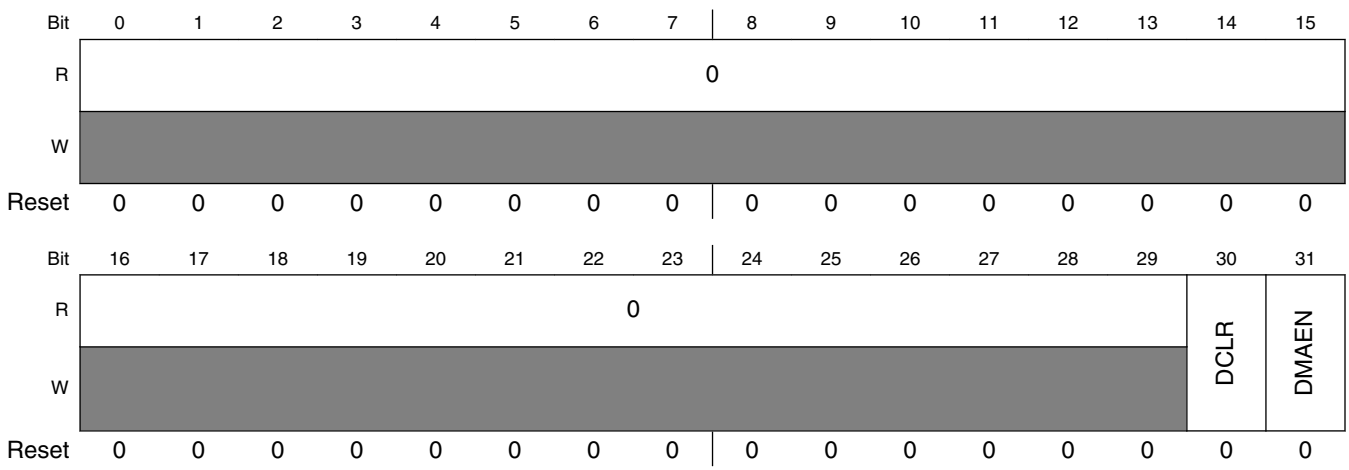
Table continues on the next page...

ADC_WTIMR field descriptions (continued)

Field	Description
28 MSKWDG1H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 1 (WD1) being higher than the programmed higher threshold. When set, the interrupt is enabled.
29 MSKWDG1L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 1 (WD1) being lower than the programmed lower threshold. When set, the interrupt is enabled.
30 MSKWDG0H	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 0 (WD0) being higher than the programmed higher threshold. When set, the interrupt is enabled.
31 MSKWDG0L	This bit enables/disables the interrupt generated by the converted value sampled from a channel monitored by Watchdog 0 (WD0) being lower than the programmed lower threshold. When set, the interrupt is enabled.

34.5.13 DMA Enable register (ADC_DMAE)

Address: 0h base + 40h offset = 40h



ADC_DMAE field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 DCLR	DMA Clear sequence enable. 0 DMA request cleared by Acknowledge from DMA controller 1 DMA request cleared on read of data registers
31 DMAEN	DMA global enable. When this bit is set, the DMA feature is enabled.

34.5.14 DMA Channel Select Register 0 (ADC_DMAR0)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DMA31	DMA30	DMA29	DMA28	DMA27	DMA26	DMA25	DMA24	DMA23	DMA22	DMA21	DMA20	DMA19	DMA18	DMA17	DMA16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DMA15	DMA14	DMA13	DMA12	DMA11	DMA10	DMA9	DMA8	DMA7	DMA6	DMA5	DMA4	DMA3	DMA2	DMA1	DMA0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_DMAR0 field descriptions

Field	Description
0 DMA31	DMA enable for channel 31. When set, transferring data in DMA mode is enabled for channel 31.
1 DMA30	DMA enable for channel 30. When set, transferring data in DMA mode is enabled for channel 30.
2 DMA29	DMA enable for channel 29. When set, transferring data in DMA mode is enabled for channel 29.
3 DMA28	DMA enable for channel 28. When set, transferring data in DMA mode is enabled for channel 28.
4 DMA27	DMA enable for channel 27. When set, transferring data in DMA mode is enabled for channel 27.
5 DMA26	DMA enable for channel 26. When set, transferring data in DMA mode is enabled for channel 26.
6 DMA25	DMA enable for channel 25. When set, transferring data in DMA mode is enabled for channel 25.

Table continues on the next page...

ADC_DMAR0 field descriptions (continued)

Field	Description
7 DMA24	DMA enable for channel 24. When set, transferring data in DMA mode is enabled for channel 24.
8 DMA23	DMA enable for channel 23. When set, transferring data in DMA mode is enabled for channel 23.
9 DMA22	DMA enable for channel 22. When set, transferring data in DMA mode is enabled for channel 22.
10 DMA21	DMA enable for channel 21. When set, transferring data in DMA mode is enabled for channel 21.
11 DMA20	DMA enable for channel 20. When set, transferring data in DMA mode is enabled for channel 20.
12 DMA19	DMA enable for channel 19. When set, transferring data in DMA mode is enabled for channel 19.
13 DMA18	DMA enable for channel 18. When set, transferring data in DMA mode is enabled for channel 18.
14 DMA17	DMA enable for channel 17. When set, transferring data in DMA mode is enabled for channel 17.
15 DMA16	DMA enable for channel 16. When set, transferring data in DMA mode is enabled for channel 16.
16 DMA15	DMA enable for channel 15. When set, transferring data in DMA mode is enabled for channel 15.
17 DMA14	DMA enable for channel 14. When set, transferring data in DMA mode is enabled for channel 14.
18 DMA13	DMA enable for channel 13. When set, transferring data in DMA mode is enabled for channel 13.
19 DMA12	DMA enable for channel 12. When set, transferring data in DMA mode is enabled for channel 12.
20 DMA11	DMA enable for channel 11. When set, transferring data in DMA mode is enabled for channel 11.
21 DMA10	DMA enable for channel 10. When set, transferring data in DMA mode is enabled for channel 10.
22 DMA9	DMA enable for channel 9. When set, transferring data in DMA mode is enabled for channel 9.
23 DMA8	DMA enable for channel 8. When set, transferring data in DMA mode is enabled for channel 8.
24 DMA7	DMA enable for channel 7.

Table continues on the next page...

ADC_DMAR0 field descriptions (continued)

Field	Description
	When set, transferring data in DMA mode is enabled for channel 7.
25 DMA6	DMA enable for channel 6. When set, transferring data in DMA mode is enabled for channel 6.
26 DMA5	DMA enable for channel 5. When set, transferring data in DMA mode is enabled for channel 5.
27 DMA4	DMA enable for channel 4. When set, transferring data in DMA mode is enabled for channel 4.
28 DMA3	DMA enable for channel 3. When set, transferring data in DMA mode is enabled for channel 3.
29 DMA2	DMA enable for channel 2. When set, transferring data in DMA mode is enabled for channel 2.
30 DMA1	DMA enable for channel 1. When set, transferring data in DMA mode is enabled for channel 1.
31 DMA0	DMA enable for channel 0. When set, transferring data in DMA mode is enabled for channel 0.

34.5.15 DMA Channel Select Register 1 (ADC_DMAR1)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DMA63	DMA62	DMA61	DMA60	DMA59	DMA58	DMA57	DMA56	DMA55	DMA54	DMA53	DMA52	DMA51	DMA50	DMA49	DMA48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DMA47	DMA46	DMA45	DMA44	DMA43	DMA42	DMA41	DMA40	DMA39	DMA38	DMA37	DMA36	DMA35	DMA34	DMA33	DMA32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_DMAR1 field descriptions

Field	Description
0 DMA63	DMA enable for channel 63. When set, transferring data in DMA mode is enabled for channel 63.
1 DMA62	DMA enable for channel 62. When set, transferring data in DMA mode is enabled for channel 62.
2 DMA61	DMA enable for channel 61. When set, transferring data in DMA mode is enabled for channel 61.
3 DMA60	DMA enable for channel 60. When set, transferring data in DMA mode is enabled for channel 60.
4 DMA59	DMA enable for channel 59. When set, transferring data in DMA mode is enabled for channel 59.
5 DMA58	DMA enable for channel 58. When set, transferring data in DMA mode is enabled for channel 58.
6 DMA57	DMA enable for channel 57. When set, transferring data in DMA mode is enabled for channel 57.
7 DMA56	DMA enable for channel 56. When set, transferring data in DMA mode is enabled for channel 56.
8 DMA55	DMA enable for channel 55. When set, transferring data in DMA mode is enabled for channel 55.
9 DMA54	DMA enable for channel 54. When set, transferring data in DMA mode is enabled for channel 54.
10 DMA53	DMA enable for channel 53. When set, transferring data in DMA mode is enabled for channel 53.
11 DMA52	DMA enable for channel 52. When set, transferring data in DMA mode is enabled for channel 52.
12 DMA51	DMA enable for channel 51. When set, transferring data in DMA mode is enabled for channel 51.
13 DMA50	DMA enable for channel 50. When set, transferring data in DMA mode is enabled for channel 50.
14 DMA49	DMA enable for channel 49. When set, transferring data in DMA mode is enabled for channel 49.
15 DMA48	DMA enable for channel 48. When set, transferring data in DMA mode is enabled for channel 48.
16 DMA47	DMA enable for channel 47. When set, transferring data in DMA mode is enabled for channel 47.
17 DMA46	DMA enable for channel 46.

Table continues on the next page...

ADC_DMAR1 field descriptions (continued)

Field	Description
	When set, transferring data in DMA mode is enabled for channel 46.
18 DMA45	DMA enable for channel 45. When set, transferring data in DMA mode is enabled for channel 45.
19 DMA44	DMA enable for channel 44. When set, transferring data in DMA mode is enabled for channel 44.
20 DMA43	DMA enable for channel 43. When set, transferring data in DMA mode is enabled for channel 43.
21 DMA42	DMA enable for channel 42. When set, transferring data in DMA mode is enabled for channel 42.
22 DMA41	DMA enable for channel 41. When set, transferring data in DMA mode is enabled for channel 41.
23 DMA40	DMA enable for channel 40. When set, transferring data in DMA mode is enabled for channel 40.
24 DMA39	DMA enable for channel 39. When set, transferring data in DMA mode is enabled for channel 39.
25 DMA38	DMA enable for channel 38. When set, transferring data in DMA mode is enabled for channel 38.
26 DMA37	DMA enable for channel 37. When set, transferring data in DMA mode is enabled for channel 37.
27 DMA36	DMA enable for channel 36. When set, transferring data in DMA mode is enabled for channel 36.
28 DMA35	DMA enable for channel 35. When set, transferring data in DMA mode is enabled for channel 35.
29 DMA34	DMA enable for channel 34. When set, transferring data in DMA mode is enabled for channel 34.
30 DMA33	DMA enable for channel 33. When set, transferring data in DMA mode is enabled for channel 33.
31 DMA32	DMA enable for channel 32. When set, transferring data in DMA mode is enabled for channel 32.

34.5.16 DMA Channel Select Register 2 (ADC_DMAR2)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DMA95	DMA94	DMA93	DMA92	DMA91	DMA90	DMA89	DMA88	DMA87	DMA86	DMA85	DMA84	DMA83	DMA82	DMA81	DMA80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DMA79	DMA78	DMA77	DMA76	DMA75	DMA74	DMA73	DMA72	DMA71	DMA70	DMA69	DMA68	DMA67	DMA66	DMA65	DMA64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_DMAR2 field descriptions

Field	Description
0 DMA95	DMA enable for channel 95. When set, transferring data in DMA mode is enabled for channel 95.
1 DMA94	DMA enable for channel 94. When set, transferring data in DMA mode is enabled for channel 94.
2 DMA93	DMA enable for channel 93. When set, transferring data in DMA mode is enabled for channel 93.
3 DMA92	DMA enable for channel 92. When set, transferring data in DMA mode is enabled for channel 92.
4 DMA91	DMA enable for channel 91. When set, transferring data in DMA mode is enabled for channel 91.
5 DMA90	DMA enable for channel 90. When set, transferring data in DMA mode is enabled for channel 90.
6 DMA89	DMA enable for channel 89. When set, transferring data in DMA mode is enabled for channel 89.

Table continues on the next page...

ADC_DMAR2 field descriptions (continued)

Field	Description
7 DMA88	DMA enable for channel 88. When set, transferring data in DMA mode is enabled for channel 88.
8 DMA87	DMA enable for channel 87. When set, transferring data in DMA mode is enabled for channel 87.
9 DMA86	DMA enable for channel 86. When set, transferring data in DMA mode is enabled for channel 86.
10 DMA85	DMA enable for channel 85. When set, transferring data in DMA mode is enabled for channel 85.
11 DMA84	DMA enable for channel 84. When set, transferring data in DMA mode is enabled for channel 84.
12 DMA83	DMA enable for channel 83. When set, transferring data in DMA mode is enabled for channel 83.
13 DMA82	DMA enable for channel 82. When set, transferring data in DMA mode is enabled for channel 82.
14 DMA81	DMA enable for channel 81. When set, transferring data in DMA mode is enabled for channel 81.
15 DMA80	DMA enable for channel 80. When set, transferring data in DMA mode is enabled for channel 80.
16 DMA79	DMA enable for channel 79. When set, transferring data in DMA mode is enabled for channel 79.
17 DMA78	DMA enable for channel 78. When set, transferring data in DMA mode is enabled for channel 78.
18 DMA77	DMA enable for channel 77. When set, transferring data in DMA mode is enabled for channel 77.
19 DMA76	DMA enable for channel 76. When set, transferring data in DMA mode is enabled for channel 76.
20 DMA75	DMA enable for channel 75. When set, transferring data in DMA mode is enabled for channel 75.
21 DMA74	DMA enable for channel 74. When set, transferring data in DMA mode is enabled for channel 74.
22 DMA73	DMA enable for channel 73. When set, transferring data in DMA mode is enabled for channel 73.
23 DMA72	DMA enable for channel 72. When set, transferring data in DMA mode is enabled for channel 72.
24 DMA71	DMA enable for channel 71.

Table continues on the next page...

ADC_DMAR2 field descriptions (continued)

Field	Description
	When set, transferring data in DMA mode is enabled for channel 71.
25 DMA70	DMA enable for channel 70. When set, transferring data in DMA mode is enabled for channel 70.
26 DMA69	DMA enable for channel 69. When set, transferring data in DMA mode is enabled for channel 69.
27 DMA68	DMA enable for channel 68. When set, transferring data in DMA mode is enabled for channel 68.
28 DMA67	DMA enable for channel 67. When set, transferring data in DMA mode is enabled for channel 67.
29 DMA66	DMA enable for channel 66. When set, transferring data in DMA mode is enabled for channel 66.
30 DMA65	DMA enable for channel 65. When set, transferring data in DMA mode is enabled for channel 65.
31 DMA64	DMA enable for channel 64. When set, transferring data in DMA mode is enabled for channel 64.

34.5.17 Threshold Register (ADC_THRHLR*n*)

Address: 0h base + 60h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				THRH												0				THRL											
W																																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

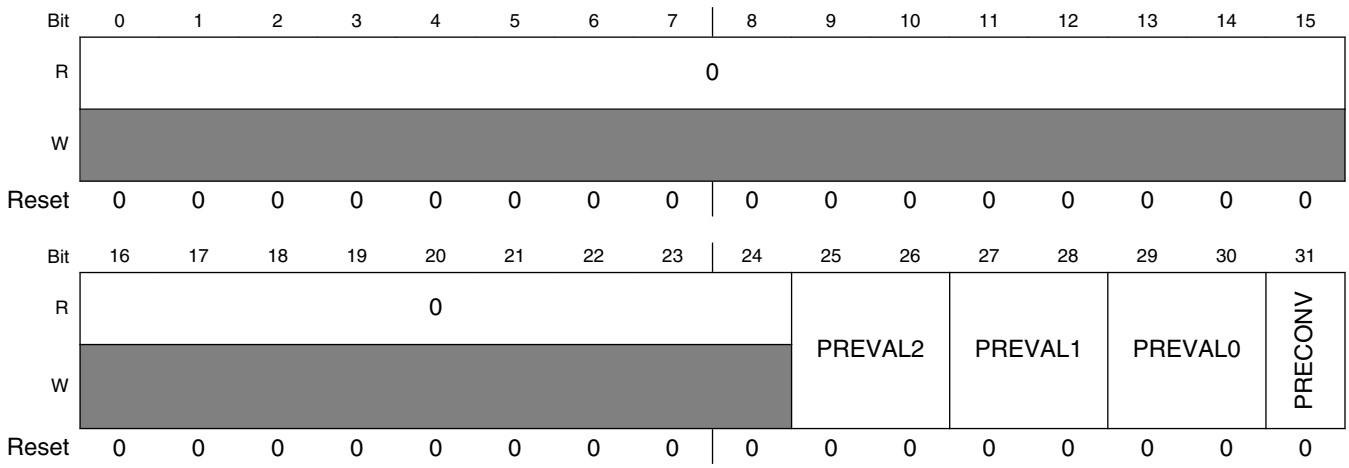
ADC_THRHLR*n* field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for watchdog n.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for watchdog n.

34.5.18 Presampling Control Register (ADC_PSCR)

See the chip configuration details for the availability of this register.

Address: 0h base + 80h offset = 80h



ADC_PSCR field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–26 PREVAL2	Internal voltage selection for Presampling Selects analog input voltage for presampling from the available four internal voltages for External channels (channel 64 to 95). Refer to Presampling channel enable.
27–28 PREVAL1	Internal voltage selection for Presampling. These two bits select analog input voltage for presampling from the available internal voltages using internal extended channels (channel 32 to 63). Refer to Presampling channel enable.
29–30 PREVAL0	Internal voltage selection for Presampling. These two bits select analog input voltage for presampling from the available four internal voltages for Internal precision channels (Channel 0 to 31). Refer to Presampling channel enable.
31 PRECONV	Convert Presampled value If bit PRECONV is set, presampling is followed by the conversion. Sampling will be bypassed and conversion of presampled data will be done. This bit has no effect on conversion of a channel when presampling on that channel is disabled (for example, PSR*x*[PRES*y*] = 0 , where *x* = register index and *y* = channel number).

34.5.19 Presampling register 0 (ADC_PSR0)

See the chip configuration details for the availability of this register.

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRES31	PRES30	PRES29	PRES28	PRES27	PRES26	PRES25	PRES24	PRES23	PRES22	PRES21	PRES20	PRES19	PRES18	PRES17	PRES16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRES15	PRES14	PRES13	PRES12	PRES11	PRES10	PRES9	PRES8	PRES7	PRES6	PRES5	PRES4	PRES3	PRES2	PRES1	PRES0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_PSR0 field descriptions

Field	Description
0 PRES31	Presampling enable for channel 31. When set, presampling is enabled for channel 31.
1 PRES30	Presampling enable for channel 30. When set, presampling is enabled for channel 30.
2 PRES29	Presampling enable for channel 29. When set, presampling is enabled for channel 29.
3 PRES28	Presampling enable for channel 28. When set, presampling is enabled for channel 28.
4 PRES27	Presampling enable for channel 27. When set, presampling is enabled for channel 27.
5 PRES26	Presampling enable for channel 26. When set, presampling is enabled for channel 26.
6 PRES25	Presampling enable for channel 25. When set, presampling is enabled for channel 25.
7 PRES24	Presampling enable for channel 24. When set, presampling is enabled for channel 24.
8 PRES23	Presampling enable for channel 23.

Table continues on the next page...

ADC_PSR0 field descriptions (continued)

Field	Description
	When set, presampling is enabled for channel 23.
9 PRES22	Presampling enable for channel 22. When set, presampling is enabled for channel 22.
10 PRES21	Presampling enable for channel 21. When set, presampling is enabled for channel 21.
11 PRES20	Presampling enable for channel 20. When set, presampling is enabled for channel 20.
12 PRES19	Presampling enable for channel 19. When set, presampling is enabled for channel 19.
13 PRES18	Presampling enable for channel 18. When set, presampling is enabled for channel 18.
14 PRES17	Presampling enable for channel 17. When set, presampling is enabled for channel 17.
15 PRES16	Presampling enable for channel 16. When set, presampling is enabled for channel 16.
16 PRES15	Presampling enable for channel 15. When set, presampling is enabled for channel 15.
17 PRES14	Presampling enable for channel 14. When set, presampling is enabled for channel 14.
18 PRES13	Presampling enable for channel 13. When set, presampling is enabled for channel 13.
19 PRES12	Presampling enable for channel 12. When set, presampling is enabled for channel 12.
20 PRES11	Presampling enable for channel 11. When set, presampling is enabled for channel 11.
21 PRES10	Presampling enable for channel 10. When set, presampling is enabled for channel 10.
22 PRES9	Presampling enable for channel 9. When set, presampling is enabled for channel 9.
23 PRES8	Presampling enable for channel 8. When set, presampling is enabled for channel 8.
24 PRES7	Presampling enable for channel 7. When set, presampling is enabled for channel 7.
25 PRES6	Presampling enable for channel 6. When set, presampling is enabled for channel 6.

Table continues on the next page...

ADC_PSR0 field descriptions (continued)

Field	Description
26 PRES5	Presampling enable for channel 5. When set, presampling is enabled for channel 5.
27 PRES4	Presampling enable for channel 4. When set, presampling is enabled for channel 4.
28 PRES3	Presampling enable for channel 3. When set, presampling is enabled for channel 3.
29 PRES2	Presampling enable for channel 2. When set, presampling is enabled for channel 2.
30 PRES1	Presampling enable for channel 1. When set, presampling is enabled for channel 1.
31 PRES0	Presampling enable for channel 0. When set, presampling is enabled for channel 0.

34.5.20 Presampling register 1 (ADC_PSR1)

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRES63	PRES62	PRES61	PRES60	PRES59	PRES58	PRES57	PRES56	PRES55	PRES54	PRES53	PRES52	PRES51	PRES50	PRES49	PRES48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRES47	PRES46	PRES45	PRES44	PRES43	PRES42	PRES41	PRES40	PRES39	PRES38	PRES37	PRES36	PRES35	PRES34	PRES33	PRES32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_PSR1 field descriptions

Field	Description
0 PRES63	Presampling enable for channel 63. When set, presampling is enabled for channel 63.
1 PRES62	Presampling enable for channel 62. When set, presampling is enabled for channel 62.

Table continues on the next page...

ADC_PSR1 field descriptions (continued)

Field	Description
2 PRES61	Presampling enable for channel 61. When set, presampling is enabled for channel 61.
3 PRES60	Presampling enable for channel 60. When set, presampling is enabled for channel 60.
4 PRES59	Presampling enable for channel 59. When set, presampling is enabled for channel 59.
5 PRES58	Presampling enable for channel 58. When set, presampling is enabled for channel 58.
6 PRES57	Presampling enable for channel 57. When set, presampling is enabled for channel 57.
7 PRES56	Presampling enable for channel 56. When set, presampling is enabled for channel 56.
8 PRES55	Presampling enable for channel 55. When set, presampling is enabled for channel 55.
9 PRES54	Presampling enable for channel 54. When set, presampling is enabled for channel 54.
10 PRES53	Presampling enable for channel 53. When set, presampling is enabled for channel 53.
11 PRES52	Presampling enable for channel 52. When set, presampling is enabled for channel 52.
12 PRES51	Presampling enable for channel 51. When set, presampling is enabled for channel 51.
13 PRES50	Presampling enable for channel 50. When set, presampling is enabled for channel 50.
14 PRES49	Presampling enable for channel 49. When set, presampling is enabled for channel 49.
15 PRES48	Presampling enable for channel 48. When set, presampling is enabled for channel 48.
16 PRES47	Presampling enable for channel 47. When set, presampling is enabled for channel 47.
17 PRES46	Presampling enable for channel 46. When set, presampling is enabled for channel 46.
18 PRES45	Presampling enable for channel 45. When set, presampling is enabled for channel 45.
19 PRES44	Presampling enable for channel 44.

Table continues on the next page...

ADC_PSR1 field descriptions (continued)

Field	Description
	When set, presampling is enabled for channel 44.
20 PRES43	Presampling enable for channel 43. When set, presampling is enabled for channel 43.
21 PRES42	Presampling enable for channel 42. When set, presampling is enabled for channel 42.
22 PRES41	Presampling enable for channel 41. When set, presampling is enabled for channel 41.
23 PRES40	Presampling enable for channel 40. When set, presampling is enabled for channel 40.
24 PRES39	Presampling enable for channel 39. When set, presampling is enabled for channel 39.
25 PRES38	Presampling enable for channel 38. When set, presampling is enabled for channel 38.
26 PRES37	Presampling enable for channel 37. When set, presampling is enabled for channel 37.
27 PRES36	Presampling enable for channel 36. When set, presampling is enabled for channel 36.
28 PRES35	Presampling enable for channel 35. When set, presampling is enabled for channel 35.
29 PRES34	Presampling enable for channel 34. When set, presampling is enabled for channel 34.
30 PRES33	Presampling enable for channel 33. When set, presampling is enabled for channel 33.
31 PRES32	Presampling enable for channel 32. When set, presampling is enabled for channel 32.

34.5.21 Presampling register 2 (ADC_PSR2)

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRES95	PRES94	PRES93	PRES92	PRES91	PRES90	PRES89	PRES88	PRES87	PRES86	PRES85	PRES84	PRES83	PRES82	PRES81	PRES80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRES79	PRES78	PRES77	PRES76	PRES75	PRES74	PRES73	PRES72	PRES71	PRES70	PRES69	PRES68	PRES67	PRES66	PRES65	PRES64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_PSR2 field descriptions

Field	Description
0 PRES95	Presampling enable for channel 95. When set, presampling is enabled for channel 95.
1 PRES94	Presampling enable for channel 94. When set, presampling is enabled for channel 94.
2 PRES93	Presampling enable for channel 93. When set, presampling is enabled for channel 93.
3 PRES92	Presampling enable for channel 92. When set, presampling is enabled for channel 92.
4 PRES91	Presampling enable for channel 91. When set, presampling is enabled for channel 91.
5 PRES90	Presampling enable for channel 90. When set, presampling is enabled for channel 90.
6 PRES89	Presampling enable for channel 89. When set, presampling is enabled for channel 89.
7 PRES88	Presampling enable for channel 88. When set, presampling is enabled for channel 88.
8 PRES87	Presampling enable for channel 87. When set, presampling is enabled for channel 87.
9 PRES86	Presampling enable for channel 86. When set, presampling is enabled for channel 86.

Table continues on the next page...

ADC_PSR2 field descriptions (continued)

Field	Description
10 PRES85	Presampling enable for channel 85. When set, presampling is enabled for channel 85.
11 PRES84	Presampling enable for channel 84. When set, presampling is enabled for channel 84.
12 PRES83	Presampling enable for channel 83. When set, presampling is enabled for channel 83.
13 PRES82	Presampling enable for channel 82. When set, presampling is enabled for channel 82.
14 PRES81	Presampling enable for channel 81. When set, presampling is enabled for channel 81.
15 PRES80	Presampling enable for channel 80. When set, presampling is enabled for channel 80.
16 PRES79	Presampling enable for channel 79. When set, presampling is enabled for channel 79.
17 PRES78	Presampling enable for channel 78. When set, presampling is enabled for channel 78.
18 PRES77	Presampling enable for channel 77. When set, presampling is enabled for channel 77.
19 PRES76	Presampling enable for channel 76. When set, presampling is enabled for channel 76.
20 PRES75	Presampling enable for channel 75. When set, presampling is enabled for channel 75.
21 PRES74	Presampling enable for channel 74. When set, presampling is enabled for channel 74.
22 PRES73	Presampling enable for channel 73. When set, presampling is enabled for channel 73.
23 PRES72	Presampling enable for channel 72. When set, presampling is enabled for channel 72.
24 PRES71	Presampling enable for channel 71. When set, presampling is enabled for channel 71.
25 PRES70	Presampling enable for channel 70. When set, presampling is enabled for channel 70.
26 PRES69	Presampling enable for channel 69. When set, presampling is enabled for channel 69.
27 PRES68	Presampling enable for channel 68.

Table continues on the next page...

ADC_PSR2 field descriptions (continued)

Field	Description
	When set, presampling is enabled for channel 68.
28 PRES67	Presampling enable for channel 67. When set, presampling is enabled for channel 67.
29 PRES66	Presampling enable for channel 66. When set, presampling is enabled for channel 66.
30 PRES65	Presampling enable for channel 65. When set, presampling is enabled for channel 65.
31 PRES64	Presampling enable for channel 64. When set, presampling is enabled for channel 64.

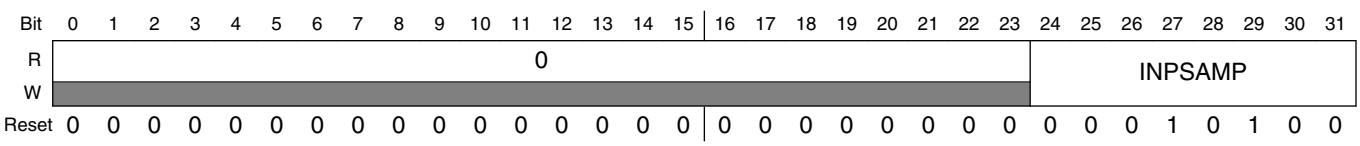
34.5.22 Conversion Timing Register 0 (ADC_CTR0)

Conversion timing register. CTR0 is associated with channels 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 94h offset = 94h



ADC_CTR0 field descriptions

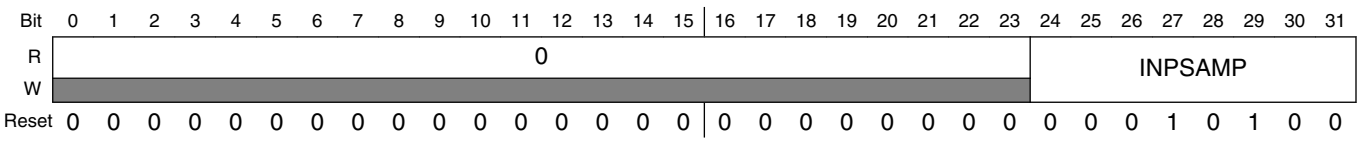
Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 INPSAMP	Configuration of sampling phase duration. This value tells the sample period duration in terms of the SAR Controller operating clock. The minimum acceptable value is 8. Setting a lower value automatically takes 8 cycles. **NOTE:** The Bandgap voltage sampling time is controlled by CTR0[INPSAMP]. CTR0[INPSAMP] is also used for the external channel sample time. When the user software needs to sample the Bandgap voltage, it must change the CTR0[INPSAMP] value to allow for a longer time while the Bandgap is being sampled, then change its value back to the normal value for sampling external channels.

34.5.23 Conversion Timing Register 1 (ADC_CTR1)

CTR1 is associated with channels 32 to 63.

Please see the chip-specific configuration information as all channels may not be available in all ADC instances.

Address: 0h base + 98h offset = 98h



ADC_CTR1 field descriptions

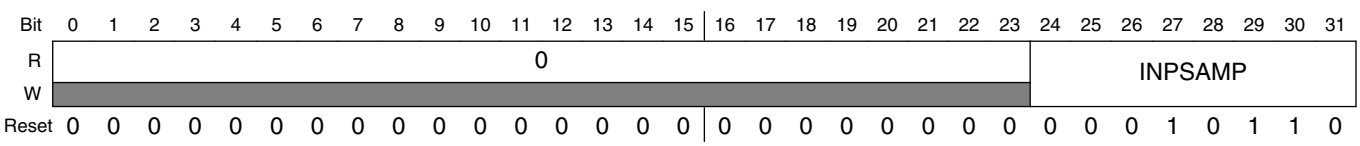
Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 INPSAMP	Configuration of sampling phase duration. This value tells the sample period duration in terms of the SAR Controller operating clock. The minimum acceptable value is 8. Setting a lower value automatically takes 8 cycles.

34.5.24 Conversion Timing Register 2 (ADC_CTR2)

CTR2 is associated to channels 64 to 95.

Please see the chip-specific configuration information as all channels may not be available in all ADC instances.

Address: 0h base + 9Ch offset = 9Ch



ADC_CTR2 field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 INPSAMP	Configuration of sampling phase duration. This value tells the sample period duration in terms of the SAR Controller operating clock. The minimum acceptable value is 8. Setting a lower value automatically takes 8 cycles.

34.5.25 Normal Conversion Mask Register 0 (ADC_NCMR0)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_NCMR0 field descriptions

Field	Description
0 CH31	Normal sampling enable for channel 31. When set, normal sampling is enabled for channel 31.
1 CH30	Normal sampling enable for channel 30. When set, normal sampling is enabled for channel 30.
2 CH29	Normal sampling enable for channel 29. When set, normal sampling is enabled for channel 29.
3 CH28	Normal sampling enable for channel 28. When set, normal sampling is enabled for channel 28.
4 CH27	Normal sampling enable for channel 27. When set, normal sampling is enabled for channel 27.
5 CH26	Normal sampling enable for channel 26. When set, normal sampling is enabled for channel 26.
6 CH25	Normal sampling enable for channel 25. When set, normal sampling is enabled for channel 25.

Table continues on the next page...

ADC_NCMR0 field descriptions (continued)

Field	Description
7 CH24	Normal sampling enable for channel 24. When set, normal sampling is enabled for channel 24.
8 CH23	Normal sampling enable for channel 23. When set, normal sampling is enabled for channel 23.
9 CH22	Normal sampling enable for channel 22. When set, normal sampling is enabled for channel 22.
10 CH21	Normal sampling enable for channel 21. When set, normal sampling is enabled for channel 21.
11 CH20	Normal sampling enable for channel 20. When set, normal sampling is enabled for channel 20.
12 CH19	Normal sampling enable for channel 19. When set, normal sampling is enabled for channel 19.
13 CH18	Normal sampling enable for channel 18. When set, normal sampling is enabled for channel 18.
14 CH17	Normal sampling enable for channel 17. When set, normal sampling is enabled for channel 17.
15 CH16	Normal sampling enable for channel 16. When set, normal sampling is enabled for channel 16.
16 CH15	Normal sampling enable for channel 15. When set, normal sampling is enabled for channel 15.
17 CH14	Normal sampling enable for channel 14. When set, normal sampling is enabled for channel 14.
18 CH13	Normal sampling enable for channel 13. When set, normal sampling is enabled for channel 13.
19 CH12	Normal sampling enable for channel 12. When set, normal sampling is enabled for channel 12.
20 CH11	Normal sampling enable for channel 11. When set, normal sampling is enabled for channel 11.
21 CH10	Normal sampling enable for channel 10. When set, normal sampling is enabled for channel 10.
22 CH9	Normal sampling enable for channel 9. When set, normal sampling is enabled for channel 9.
23 CH8	Normal sampling enable for channel 8. When set, normal sampling is enabled for channel 8.
24 CH7	Normal sampling enable for channel 7.

Table continues on the next page...

ADC_NCMR0 field descriptions (continued)

Field	Description
	When set, normal sampling is enabled for channel 7.
25 CH6	Normal sampling enable for channel 6. When set, normal sampling is enabled for channel 6.
26 CH5	Normal sampling enable for channel 5. When set, normal sampling is enabled for channel 5.
27 CH4	Normal sampling enable for channel 4. When set, normal sampling is enabled for channel 4.
28 CH3	Normal sampling enable for channel 3. When set, normal sampling is enabled for channel 3.
29 CH2	Normal sampling enable for channel 2. When set, normal sampling is enabled for channel 2.
30 CH1	Normal sampling enable for channel 1. When set, normal sampling is enabled for channel 1.
31 CH0	Normal sampling enable for channel 0. When set, normal sampling is enabled for channel 0.

34.5.26 Normal Conversion Mask Register 1 (ADC_NCMR1)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CH63	CH62	CH61	CH60	CH59	CH58	CH57	CH56	CH55	CH54	CH53	CH52	CH51	CH50	CH49	CH48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CH47	CH46	CH45	CH44	CH43	CH42	CH41	CH40	CH39	CH38	CH37	CH36	CH35	CH34	CH33	CH32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_NCMR1 field descriptions

Field	Description
0 CH63	Normal sampling enable for channel 63. When set, normal sampling is enabled for channel 63.
1 CH62	Normal sampling enable for channel 62. When set, normal sampling is enabled for channel 62.
2 CH61	Normal sampling enable for channel 61. When set, normal sampling is enabled for channel 61.
3 CH60	Normal sampling enable for channel 60. When set, normal sampling is enabled for channel 60.
4 CH59	Normal sampling enable for channel 59. When set, normal sampling is enabled for channel 59.
5 CH58	Normal sampling enable for channel 58. When set, normal sampling is enabled for channel 58.
6 CH57	Normal sampling enable for channel 57. When set, normal sampling is enabled for channel 57.
7 CH56	Normal sampling enable for channel 56. When set, normal sampling is enabled for channel 56.
8 CH55	Normal sampling enable for channel 55. When set, normal sampling is enabled for channel 55.
9 CH54	Normal sampling enable for channel 54. When set, normal sampling is enabled for channel 54.
10 CH53	Normal sampling enable for channel 53. When set, normal sampling is enabled for channel 53.
11 CH52	Normal sampling enable for channel 52. When set, normal sampling is enabled for channel 52.
12 CH51	Normal sampling enable for channel 51. When set, normal sampling is enabled for channel 51.
13 CH50	Normal sampling enable for channel 50. When set, normal sampling is enabled for channel 50.
14 CH49	Normal sampling enable for channel 49. When set, normal sampling is enabled for channel 49.
15 CH48	Normal sampling enable for channel 48. When set, normal sampling is enabled for channel 48.
16 CH47	Normal sampling enable for channel 47. When set, normal sampling is enabled for channel 47.
17 CH46	Normal sampling enable for channel 46.

Table continues on the next page...

ADC_NCMR1 field descriptions (continued)

Field	Description
	When set, normal sampling is enabled for channel 46.
18 CH45	Normal sampling enable for channel 45. When set, normal sampling is enabled for channel 45.
19 CH44	Normal sampling enable for channel 44. When set, normal sampling is enabled for channel 44.
20 CH43	Normal sampling enable for channel 43. When set, normal sampling is enabled for channel 43.
21 CH42	Normal sampling enable for channel 42. When set, normal sampling is enabled for channel 42.
22 CH41	Normal sampling enable for channel 41. When set, normal sampling is enabled for channel 41.
23 CH40	Normal sampling enable for channel 40. When set, normal sampling is enabled for channel 40.
24 CH39	Normal sampling enable for channel 39. When set, normal sampling is enabled for channel 39.
25 CH38	Normal sampling enable for channel 38. When set, normal sampling is enabled for channel 38.
26 CH37	Normal sampling enable for channel 37. When set, normal sampling is enabled for channel 37.
27 CH36	Normal sampling enable for channel 36. When set, normal sampling is enabled for channel 36.
28 CH35	Normal sampling enable for channel 35. When set, normal sampling is enabled for channel 35.
29 CH34	Normal sampling enable for channel 34. When set, normal sampling is enabled for channel 34.
30 CH33	Normal sampling enable for channel 33. When set, normal sampling is enabled for channel 33.
31 CH32	Normal sampling enable for channel 32. When set, normal sampling is enabled for channel 32.

34.5.27 Normal Conversion Mask Register 2 (ADC_NCMR2)

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + ACh offset = ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CH95	CH94	CH93	CH92	CH91	CH90	CH89	CH88	CH87	CH86	CH85	CH84	CH83	CH82	CH81	CH80
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH79	CH78	CH77	CH76	CH75	CH74	CH73	CH72	CH71	CH70	CH69	CH68	CH67	CH66	CH65	CH64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_NCMR2 field descriptions

Field	Description
0 CH95	Normal sampling enable for channel 95. When set, normal sampling is enabled for channel 95.
1 CH94	Normal sampling enable for channel 94. When set, normal sampling is enabled for channel 94.
2 CH93	Normal sampling enable for channel 93. When set, normal sampling is enabled for channel 93.
3 CH92	Normal sampling enable for channel 92. When set, normal sampling is enabled for channel 92.
4 CH91	Normal sampling enable for channel 91. When set, normal sampling is enabled for channel 91.
5 CH90	Normal sampling enable for channel 90. When set, normal sampling is enabled for channel 90.
6 CH89	Normal sampling enable for channel 89. When set, normal sampling is enabled for channel 89.

Table continues on the next page...

ADC_NCMR2 field descriptions (continued)

Field	Description
7 CH88	Normal sampling enable for channel 88. When set, normal sampling is enabled for channel 88.
8 CH87	Normal sampling enable for channel 87. When set, normal sampling is enabled for channel 87.
9 CH86	Normal sampling enable for channel 86. When set, normal sampling is enabled for channel 86.
10 CH85	Normal sampling enable for channel 85. When set, normal sampling is enabled for channel 85.
11 CH84	Normal sampling enable for channel 84. When set, normal sampling is enabled for channel 84.
12 CH83	Normal sampling enable for channel 83. When set, normal sampling is enabled for channel 83.
13 CH82	Normal sampling enable for channel 82. When set, normal sampling is enabled for channel 82.
14 CH81	Normal sampling enable for channel 81. When set, normal sampling is enabled for channel 81.
15 CH80	Normal sampling enable for channel 80. When set, normal sampling is enabled for channel 80.
16 CH79	Normal sampling enable for channel 79. When set, normal sampling is enabled for channel 79.
17 CH78	Normal sampling enable for channel 78. When set, normal sampling is enabled for channel 78.
18 CH77	Normal sampling enable for channel 77. When set, normal sampling is enabled for channel 77.
19 CH76	Normal sampling enable for channel 76. When set, normal sampling is enabled for channel 76.
20 CH75	Normal sampling enable for channel 75. When set, normal sampling is enabled for channel 75.
21 CH74	Normal sampling enable for channel 74. When set, normal sampling is enabled for channel 74.
22 CH73	Normal sampling enable for channel 73. When set, normal sampling is enabled for channel 73.
23 CH72	Normal sampling enable for channel 72. When set, normal sampling is enabled for channel 72.
24 CH71	Normal sampling enable for channel 71.

Table continues on the next page...

ADC_NCMR2 field descriptions (continued)

Field	Description
	When set, normal sampling is enabled for channel 71.
25 CH70	Normal sampling enable for channel 70. When set, normal sampling is enabled for channel 70.
26 CH69	Normal sampling enable for channel 69. When set, normal sampling is enabled for channel 69.
27 CH68	Normal sampling enable for channel 68. When set, normal sampling is enabled for channel 68.
28 CH67	Normal sampling enable for channel 67. When set, normal sampling is enabled for channel 67.
29 CH66	Normal sampling enable for channel 66. When set, normal sampling is enabled for channel 66.
30 CH65	Normal sampling enable for channel 65. When set, normal sampling is enabled for channel 65.
31 CH64	Normal sampling enable for channel 64. When set, normal sampling is enabled for channel 64.

34.5.28 Injected Conversion Mask Register 0 (ADC_JCMR0)

Enable bits of Injected Sampling for channel 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CH31	CH30	CH29	CH28	CH27	CH26	CH25	CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_JCMR0 field descriptions

Field	Description
0 CH31	Injected sampling enable for channel 31. When set, injected sampling is enabled for channel 31.
1 CH30	Injected sampling enable for channel 30. When set, injected sampling is enabled for channel 30.
2 CH29	Injected sampling enable for channel 29. When set, injected sampling is enabled for channel 29.
3 CH28	Injected sampling enable for channel 28. When set, injected sampling is enabled for channel 28.
4 CH27	Injected sampling enable for channel 27. When set, injected sampling is enabled for channel 27.
5 CH26	Injected sampling enable for channel 26. When set, injected sampling is enabled for channel 26.
6 CH25	Injected sampling enable for channel 25. When set, injected sampling is enabled for channel 25.
7 CH24	Injected sampling enable for channel 24. When set, injected sampling is enabled for channel 24.
8 CH23	Injected sampling enable for channel 23. When set, injected sampling is enabled for channel 23.
9 CH22	Injected sampling enable for channel 22. When set, injected sampling is enabled for channel 22.
10 CH21	Injected sampling enable for channel 21. When set, injected sampling is enabled for channel 21.
11 CH20	Injected sampling enable for channel 20. When set, injected sampling is enabled for channel 20.
12 CH19	Injected sampling enable for channel 19. When set, injected sampling is enabled for channel 19.
13 CH18	Injected sampling enable for channel 18. When set, injected sampling is enabled for channel 18.
14 CH17	Injected sampling enable for channel 17. When set, injected sampling is enabled for channel 17.
15 CH16	Injected sampling enable for channel 16. When set, injected sampling is enabled for channel 16.
16 CH15	Injected sampling enable for channel 15. When set, injected sampling is enabled for channel 15.
17 CH14	Injected sampling enable for channel 14.

Table continues on the next page...

ADC_JCMR0 field descriptions (continued)

Field	Description
	When set, injected sampling is enabled for channel 14.
18 CH13	Injected sampling enable for channel 13. When set, injected sampling is enabled for channel 13.
19 CH12	Injected sampling enable for channel 12. When set, injected sampling is enabled for channel 12.
20 CH11	Injected sampling enable for channel 11. When set, injected sampling is enabled for channel 11.
21 CH10	Injected sampling enable for channel 10. When set, injected sampling is enabled for channel 10.
22 CH9	Injected sampling enable for channel 9. When set, injected sampling is enabled for channel 9.
23 CH8	Injected sampling enable for channel 8. When set, injected sampling is enabled for channel 8.
24 CH7	Injected sampling enable for channel 7. When set, injected sampling is enabled for channel 7.
25 CH6	Injected sampling enable for channel 6. When set, injected sampling is enabled for channel 6.
26 CH5	Injected sampling enable for channel 5. When set, injected sampling is enabled for channel 5.
27 CH4	Injected sampling enable for channel 4. When set, injected sampling is enabled for channel 4.
28 CH3	Injected sampling enable for channel 3. When set, injected sampling is enabled for channel 3.
29 CH2	Injected sampling enable for channel 2. When set, injected sampling is enabled for channel 2.
30 CH1	Injected sampling enable for channel 1. When set, injected sampling is enabled for channel 1.
31 CH0	Injected sampling enable for channel 0. When set, injected sampling is enabled for channel 0.

34.5.29 Injected Conversion Mask Register 1 (ADC_JCMR1)

Enable bits of Injected Sampling for channel 32 to 63.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CH63	CH62	CH61	CH60	CH59	CH58	CH57	CH56	CH55	CH54	CH53	CH52	CH51	CH50	CH49	CH48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CH47	CH46	CH45	CH44	CH43	CH42	CH41	CH40	CH39	CH38	CH37	CH36	CH35	CH34	CH33	CH32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_JCMR1 field descriptions

Field	Description
0 CH63	Injected sampling enable for channel 63. When set, injected sampling is enabled for channel 63.
1 CH62	Injected sampling enable for channel 62. When set, injected sampling is enabled for channel 62.
2 CH61	Injected sampling enable for channel 61. When set, injected sampling is enabled for channel 61.
3 CH60	Injected sampling enable for channel 60. When set, injected sampling is enabled for channel 60.
4 CH59	Injected sampling enable for channel 59. When set, injected sampling is enabled for channel 59.
5 CH58	Injected sampling enable for channel 58. When set, injected sampling is enabled for channel 58.
6 CH57	Injected sampling enable for channel 57. When set, injected sampling is enabled for channel 57.
7 CH56	Injected sampling enable for channel 56. When set, injected sampling is enabled for channel 56.
8 CH55	Injected sampling enable for channel 55. When set, injected sampling is enabled for channel 55.

Table continues on the next page...

ADC_JCMR1 field descriptions (continued)

Field	Description
9 CH54	Injected sampling enable for channel 54. When set, injected sampling is enabled for channel 54.
10 CH53	Injected sampling enable for channel 53. When set, injected sampling is enabled for channel 53.
11 CH52	Injected sampling enable for channel 52. When set, injected sampling is enabled for channel 52.
12 CH51	Injected sampling enable for channel 51. When set, injected sampling is enabled for channel 51.
13 CH50	Injected sampling enable for channel 50. When set, injected sampling is enabled for channel 50.
14 CH49	Injected sampling enable for channel 49. When set, injected sampling is enabled for channel 49.
15 CH48	Injected sampling enable for channel 48. When set, injected sampling is enabled for channel 48.
16 CH47	Injected sampling enable for channel 47. When set, injected sampling is enabled for channel 47.
17 CH46	Injected sampling enable for channel 46. When set, injected sampling is enabled for channel 46.
18 CH45	Injected sampling enable for channel 45. When set, injected sampling is enabled for channel 45.
19 CH44	Injected sampling enable for channel 44. When set, injected sampling is enabled for channel 44.
20 CH43	Injected sampling enable for channel 43. When set, injected sampling is enabled for channel 43.
21 CH42	Injected sampling enable for channel 42. When set, injected sampling is enabled for channel 42.
22 CH41	Injected sampling enable for channel 41. When set, injected sampling is enabled for channel 41.
23 CH40	Injected sampling enable for channel 40. When set, injected sampling is enabled for channel 40.
24 CH39	Injected sampling enable for channel 39. When set, injected sampling is enabled for channel 39.
25 CH38	Injected sampling enable for channel 38. When set, injected sampling is enabled for channel 38.
26 CH37	Injected sampling enable for channel 37.

Table continues on the next page...

ADC_JCMR1 field descriptions (continued)

Field	Description
	When set, injected sampling is enabled for channel 37.
27 CH36	Injected sampling enable for channel 36. When set, injected sampling is enabled for channel 36.
28 CH35	Injected sampling enable for channel 35. When set, injected sampling is enabled for channel 35.
29 CH34	Injected sampling enable for channel 34. When set, injected sampling is enabled for channel 34.
30 CH33	Injected sampling enable for channel 33. When set, injected sampling is enabled for channel 33.
31 CH32	Injected sampling enable for channel 32. When set, injected sampling is enabled for channel 32.

34.5.30 Injected Conversion Mask Register 2 (ADC_JCMR2)

Enable bits of Injected Sampling for channel 64 to 95.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CH95	CH94	CH93	CH92	CH91	CH90	CH89	CH88	CH87	CH86	CH85	CH84	CH83	CH82	CH81	CH80
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CH79	CH78	CH77	CH76	CH75	CH74	CH73	CH72	CH71	CH70	CH69	CH68	CH67	CH66	CH65	CH64
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_JCMR2 field descriptions

Field	Description
0 CH95	Injected sampling enable for channel 95. When set, injected sampling is enabled for channel 95.
1 CH94	Injected sampling enable for channel 94. When set, injected sampling is enabled for channel 94.
2 CH93	Injected sampling enable for channel 93. When set, injected sampling is enabled for channel 93.
3 CH92	Injected sampling enable for channel 92. When set, injected sampling is enabled for channel 92.
4 CH91	Injected sampling enable for channel 91. When set, injected sampling is enabled for channel 91.
5 CH90	Injected sampling enable for channel 90. When set, injected sampling is enabled for channel 90.
6 CH89	Injected sampling enable for channel 89. When set, injected sampling is enabled for channel 89.
7 CH88	Injected sampling enable for channel 88. When set, injected sampling is enabled for channel 88.
8 CH87	Injected sampling enable for channel 87. When set, injected sampling is enabled for channel 87.
9 CH86	Injected sampling enable for channel 86. When set, injected sampling is enabled for channel 86.
10 CH85	Injected sampling enable for channel 85. When set, injected sampling is enabled for channel 85.
11 CH84	Injected sampling enable for channel 84. When set, injected sampling is enabled for channel 84.
12 CH83	Injected sampling enable for channel 83. When set, injected sampling is enabled for channel 83.
13 CH82	Injected sampling enable for channel 82. When set, injected sampling is enabled for channel 82.
14 CH81	Injected sampling enable for channel 81. When set, injected sampling is enabled for channel 81.
15 CH80	Injected sampling enable for channel 80. When set, injected sampling is enabled for channel 80.
16 CH79	Injected sampling enable for channel 79. When set, injected sampling is enabled for channel 79.
17 CH78	Injected sampling enable for channel 78.

Table continues on the next page...

ADC_JCMR2 field descriptions (continued)

Field	Description
	When set, injected sampling is enabled for channel 78.
18 CH77	Injected sampling enable for channel 77. When set, injected sampling is enabled for channel 77.
19 CH76	Injected sampling enable for channel 76. When set, injected sampling is enabled for channel 76.
20 CH75	Injected sampling enable for channel 75. When set, injected sampling is enabled for channel 75.
21 CH74	Injected sampling enable for channel 74. When set, injected sampling is enabled for channel 74.
22 CH73	Injected sampling enable for channel 73. When set, injected sampling is enabled for channel 73.
23 CH72	Injected sampling enable for channel 72. When set, injected sampling is enabled for channel 72.
24 CH71	Injected sampling enable for channel 71. When set, injected sampling is enabled for channel 71.
25 CH70	Injected sampling enable for channel 70. When set, injected sampling is enabled for channel 70.
26 CH69	Injected sampling enable for channel 69. When set, injected sampling is enabled for channel 69.
27 CH68	Injected sampling enable for channel 68. When set, injected sampling is enabled for channel 68.
28 CH67	Injected sampling enable for channel 67. When set, injected sampling is enabled for channel 67.
29 CH66	Injected sampling enable for channel 66. When set, injected sampling is enabled for channel 66.
30 CH65	Injected sampling enable for channel 65. When set, injected sampling is enabled for channel 65.
31 CH64	Injected sampling enable for channel 64. When set, injected sampling is enabled for channel 64.

34.5.31 Decode Signals Delay Register (ADC_DSDR)

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																DSD															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_DSDR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 DSD	The delay between the external decode signals and the beginning of the sampling phase. It is used to take into account the settling time of the external mux. The decode signal delay is calculated as (DSD x 1/ Frequency of System Clock).

34.5.32 Power Down Exit Delay Register (ADC_PDEDR)

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								PDED							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_PDEDR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 PDED	The delay between the power down bit reset and the start of conversion. The power down delay is calculated as (PDED x 1/[ADC_clock_frequency]).

34.5.33 Channel Data Register *n* (Precision Channels) (ADC_CDR*n*)

Note

It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

Address: 0h base + 100h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CDR*n* field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.

Table continues on the next page...

ADC_CDR*n* field descriptions (continued)

Field	Description
13 OVERW	Overwrite data Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the OWREN bit of MCR register.
14–15 RESULT	These two bits reflect the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of BCTU conversion mode 11 Reserved
16–31 CDATA	Converted Data 11:0. **Note:** It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

34.5.34 Channel Data Register *n* (Internal Channels) (ADC_CDR*n*)

Note

It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

Address: 0h base + 180h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CDR*n* field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.

Table continues on the next page...

ADC_CDR*n* field descriptions (continued)

Field	Description
13 OVERW	Overwrite data Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the OWREN bit of MCR register.
14–15 RESULT	These two bits reflect the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of BCTU conversion mode 11 Reserved
16–31 CDATA	Converted Data 11:0. **Note:** It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

34.5.35 Channel Data Register *n* (External Channels) (ADC_CDR*n*)

Note

It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

Address: 0h base + 200h offset + (4d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												VALID	OVERW	RESULT	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CDR*n* field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 VALID	Used to notify when the data is valid (a new value has been written). It is automatically cleared when data is read.

Table continues on the next page...

ADC_CDR*n* field descriptions (continued)

Field	Description
13 OVERW	Overwrite data Used to notify when a conversion data is overwritten by a newer result. The new data is written or discarded according to the OWREN bit of MCR register.
14–15 RESULT	These two bits reflect the mode of conversion for the corresponding channel. 00 Data is a result of Normal conversion mode 01 Data is a result of Injected conversion mode 10 Data is a result of BCTU conversion mode 11 Reserved
16–31 CDATA	Converted Data 11:0. **Note:** It is important to note that the content of the CDATA field is affected by the setting of the ADC_MCR[WLSIDE] field, which controls whether the data is left aligned or right aligned. When ADC_MCR[WLSIDE]=1 (data to be left aligned), the CDATA field is left aligned from bit 16 to bit (16 + resolution - 1). When ADC_MCR[WLSIDE]=0 (data to be right aligned), the CDATA field is right aligned from bit (32-resolution) to bit 31. All other fields are unaffected.

34.5.36 Threshold Register (ADC_THRHLR*n*)

Address: 0h base + 280h offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				THRH												0				THRL											
W																																
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_THRHLR*n* field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–15 THRH	High threshold value for channel x.
16–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 THRL	Low threshold value for channel x.

34.5.37 Channel Watchdog Select Register 0 (ADC_CWSELR0)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2B0h offset = 2B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH7				WSEL_CH6				WSEL_CH5				WSEL_CH4				WSEL_CH3				WSEL_CH2				WSEL_CH1				WSEL_CH0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR0 field descriptions

Field	Description
0–3 WSEL_CH7	Channel Watchdog select for channel 7. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH6	Channel Watchdog select for channel 6. See WSEL_CH7.
8–11 WSEL_CH5	Channel Watchdog select for channel 5. See WSEL_CH7.
12–15 WSEL_CH4	Channel Watchdog select for channel 4. See WSEL_CH7.
16–19 WSEL_CH3	Channel Watchdog select for channel 3. See WSEL_CH7.
20–23 WSEL_CH2	Channel Watchdog select for channel 2. See WSEL_CH7.
24–27 WSEL_CH1	Channel Watchdog select for channel 1. See WSEL_CH7.
28–31 WSEL_CH0	Channel Watchdog select for channel 0. See WSEL_CH7.

34.5.38 Channel Watchdog Select Register 1 (ADC_CWSELR1)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2B4h offset = 2B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH15				WSEL_CH14				WSEL_CH13				WSEL_CH12				WSEL_CH11				WSEL_CH10				WSEL_CH9				WSEL_CH8			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR1 field descriptions

Field	Description
0–3 WSEL_CH15	Channel Watchdog select for channel 15. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH14	Channel Watchdog select for channel 14. See WSEL_CH15.
8–11 WSEL_CH13	Channel Watchdog select for channel 13. See WSEL_CH15.
12–15 WSEL_CH12	Channel Watchdog select for channel 12. See WSEL_CH15.
16–19 WSEL_CH11	Channel Watchdog select for channel 11. See WSEL_CH15.
20–23 WSEL_CH10	Channel Watchdog select for channel 10. See WSEL_CH15.
24–27 WSEL_CH9	Channel Watchdog select for channel 9. See WSEL_CH15.
28–31 WSEL_CH8	Channel Watchdog select for channel 8. See WSEL_CH15.

34.5.39 Channel Watchdog Select Register 2 (ADC_CWSELR2)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2B8h offset = 2B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH23				WSEL_CH22				WSEL_CH21				WSEL_CH20				WSEL_CH19				WSEL_CH18				WSEL_CH17				WSEL_CH16			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR2 field descriptions

Field	Description
0–3 WSEL_CH23	Channel Watchdog select for channel 23. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH22	Channel Watchdog select for channel 22. See WSEL_CH23.
8–11 WSEL_CH21	Channel Watchdog select for channel 21. See WSEL_CH23.
12–15 WSEL_CH20	Channel Watchdog select for channel 20. See WSEL_CH23.
16–19 WSEL_CH19	Channel Watchdog select for channel 19. See WSEL_CH23.
20–23 WSEL_CH18	Channel Watchdog select for channel 18. See WSEL_CH23.
24–27 WSEL_CH17	Channel Watchdog select for channel 17. See WSEL_CH23.
28–31 WSEL_CH16	Channel Watchdog select for channel 16. See WSEL_CH23.

34.5.40 Channel Watchdog Select Register 4 (ADC_CWSELR4)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2C0h offset = 2C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH39				WSEL_CH38				WSEL_CH37				WSEL_CH36				WSEL_CH35				WSEL_CH34				WSEL_CH33				WSEL_CH32			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR4 field descriptions

Field	Description
0–3 WSEL_CH39	Channel Watchdog select for channel 39. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH38	Channel Watchdog select for channel 38. See WSEL_CH39.
8–11 WSEL_CH37	Channel Watchdog select for channel 37. See WSEL_CH39.
12–15 WSEL_CH36	Channel Watchdog select for channel 36. See WSEL_CH39.
16–19 WSEL_CH35	Channel Watchdog select for channel 35. See WSEL_CH39.
20–23 WSEL_CH34	Channel Watchdog select for channel 34. See WSEL_CH39.
24–27 WSEL_CH33	Channel Watchdog select for channel 33. See WSEL_CH39.
28–31 WSEL_CH32	Channel Watchdog select for channel 32. See WSEL_CH39.

34.5.41 Channel Watchdog Select Register 5 (ADC_CWSELR5)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2C4h offset = 2C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH47				WSEL_CH46				WSEL_CH45				WSEL_CH44				WSEL_CH43				WSEL_CH42				WSEL_CH41				WSEL_CH40			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR5 field descriptions

Field	Description
0–3 WSEL_CH47	Channel Watchdog select for channel 47. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH46	Channel Watchdog select for channel 46. See WSEL_CH47.
8–11 WSEL_CH45	Channel Watchdog select for channel 45. See WSEL_CH47.
12–15 WSEL_CH44	Channel Watchdog select for channel 44. See WSEL_CH47.
16–19 WSEL_CH43	Channel Watchdog select for channel 43. See WSEL_CH47.
20–23 WSEL_CH42	Channel Watchdog select for channel 42. See WSEL_CH47.
24–27 WSEL_CH41	Channel Watchdog select for channel 41. See WSEL_CH47.
28–31 WSEL_CH40	Channel Watchdog select for channel 40. See WSEL_CH47.

34.5.42 Channel Watchdog Select Register 6 (ADC_CWSELR6)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2C8h offset = 2C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH55				WSEL_CH54				WSEL_CH53				WSEL_CH52				WSEL_CH51				WSEL_CH50				WSEL_CH49				WSEL_CH48			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR6 field descriptions

Field	Description
0–3 WSEL_CH55	Channel Watchdog select for channel 55. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH54	Channel Watchdog select for channel 54. See WSEL_CH55.
8–11 WSEL_CH53	Channel Watchdog select for channel 53. See WSEL_CH55.
12–15 WSEL_CH52	Channel Watchdog select for channel 52. See WSEL_CH55.
16–19 WSEL_CH51	Channel Watchdog select for channel 51. See WSEL_CH55.
20–23 WSEL_CH50	Channel Watchdog select for channel 50. See WSEL_CH55.
24–27 WSEL_CH49	Channel Watchdog select for channel 49. See WSEL_CH55.
28–31 WSEL_CH48	Channel Watchdog select for channel 48. See WSEL_CH55.

34.5.43 Channel Watchdog Select Register 7 (ADC_CWSELR7)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2CCh offset = 2CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH63				WSEL_CH62				WSEL_CH61				WSEL_CH60				WSEL_CH59				WSEL_CH58				WSEL_CH57				WSEL_CH56			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR7 field descriptions

Field	Description
0–3 WSEL_CH63	Channel Watchdog select for channel 63. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH62	Channel Watchdog select for channel 62. See WSEL_CH63.
8–11 WSEL_CH61	Channel Watchdog select for channel 61. See WSEL_CH63.
12–15 WSEL_CH60	Channel Watchdog select for channel 60. See WSEL_CH63.
16–19 WSEL_CH59	Channel Watchdog select for channel 59. See WSEL_CH63.
20–23 WSEL_CH58	Channel Watchdog select for channel 58. See WSEL_CH63.
24–27 WSEL_CH57	Channel Watchdog select for channel 57. See WSEL_CH63.
28–31 WSEL_CH56	Channel Watchdog select for channel 56. See WSEL_CH63.

34.5.44 Channel Watchdog Select Register 8 (ADC_CWSELR8)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2D0h offset = 2D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH71				WSEL_CH70				WSEL_CH69				WSEL_CH68				WSEL_CH67				WSEL_CH66				WSEL_CH65				WSEL_CH64			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR8 field descriptions

Field	Description
0–3 WSEL_CH71	Channel Watchdog select for channel 71. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH70	Channel Watchdog select for channel 70. See WSEL_CH71.
8–11 WSEL_CH69	Channel Watchdog select for channel 69. See WSEL_CH71.
12–15 WSEL_CH68	Channel Watchdog select for channel 68. See WSEL_CH71.
16–19 WSEL_CH67	Channel Watchdog select for channel 67. See WSEL_CH71.
20–23 WSEL_CH66	Channel Watchdog select for channel 66. See WSEL_CH71.
24–27 WSEL_CH65	Channel Watchdog select for channel 65. See WSEL_CH71.
28–31 WSEL_CH64	Channel Watchdog select for channel 64. See WSEL_CH71.

34.5.45 Channel Watchdog Select Register 9 (ADC_CWSELR9)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2D4h offset = 2D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH79				WSEL_CH78				WSEL_CH77				WSEL_CH76				WSEL_CH75				WSEL_CH74				WSEL_CH73				WSEL_CH72			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR9 field descriptions

Field	Description
0–3 WSEL_CH79	Channel Watchdog select for channel 79. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH78	Channel Watchdog select for channel 78. See WSEL_CH79.
8–11 WSEL_CH77	Channel Watchdog select for channel 77. See WSEL_CH79.
12–15 WSEL_CH76	Channel Watchdog select for channel 76. See WSEL_CH79.
16–19 WSEL_CH75	Channel Watchdog select for channel 75. See WSEL_CH79.
20–23 WSEL_CH74	Channel Watchdog select for channel 74. See WSEL_CH79.
24–27 WSEL_CH73	Channel Watchdog select for channel 73. See WSEL_CH79.
28–31 WSEL_CH72	Channel Watchdog select for channel 72. See WSEL_CH79.

34.5.46 Channel Watchdog Select Register 10 (ADC_CWSELR10)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2D8h offset = 2D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH87				WSEL_CH86				WSEL_CH85				WSEL_CH84				WSEL_CH83				WSEL_CH82				WSEL_CH81				WSEL_CH80			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR10 field descriptions

Field	Description
0–3 WSEL_CH87	Channel Watchdog select for channel 87. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH86	Channel Watchdog select for channel 86. See WSEL_CH87.
8–11 WSEL_CH85	Channel Watchdog select for channel 85. See WSEL_CH87.
12–15 WSEL_CH84	Channel Watchdog select for channel 84. See WSEL_CH87.
16–19 WSEL_CH83	Channel Watchdog select for channel 83. See WSEL_CH87.
20–23 WSEL_CH82	Channel Watchdog select for channel 82. See WSEL_CH87.
24–27 WSEL_CH81	Channel Watchdog select for channel 81. See WSEL_CH87.
28–31 WSEL_CH80	Channel Watchdog select for channel 80. See WSEL_CH87.

34.5.47 Channel Watchdog Select Register 11 (ADC_CWSELR11)

WSEL_CHn[3:0] bit field: Selects the threshold register which provides the values to be used for upper and lower thresholds for channel n.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2DCh offset = 2DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WSEL_CH95				WSEL_CH94				WSEL_CH93				WSEL_CH92				WSEL_CH91				WSEL_CH90				WSEL_CH89				WSEL_CH88			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWSELR11 field descriptions

Field	Description
0–3 WSEL_CH95	Channel Watchdog select for channel 95. 0000 : THRHLR0 register is selected 0001 : THRHLR1 register is selected 0010 : THRHLR2 register is selected ... 1110 : THRHLR14 register is selected 1111 : THRHLR15 register is selected
4–7 WSEL_CH94	Channel Watchdog select for channel 94. See WSEL_CH95.
8–11 WSEL_CH93	Channel Watchdog select for channel 93. See WSEL_CH95.
12–15 WSEL_CH92	Channel Watchdog select for channel 92. See WSEL_CH95.
16–19 WSEL_CH91	Channel Watchdog select for channel 91. See WSEL_CH95.
20–23 WSEL_CH90	Channel Watchdog select for channel 90. See WSEL_CH95.
24–27 WSEL_CH89	Channel Watchdog select for channel 89. See WSEL_CH95.
28–31 WSEL_CH88	Channel Watchdog select for channel 88. See WSEL_CH95.

34.5.48 Channel Watchdog Enable Register 0 (ADC_CWENR0)

Controls whether the watchdog is enabled for channels 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2E0h offset = 2E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CWEN31	CWEN30	CWEN29	CWEN28	CWEN27	CWEN26	CWEN25	CWEN24	CWEN23	CWEN22	CWEN21	CWEN20	CWEN19	CWEN18	CWEN17	CWEN16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CWEN15	CWEN14	CWEN13	CWEN12	CWEN11	CWEN10	CWEN9	CWEN8	CWEN7	CWEN6	CWEN5	CWEN4	CWEN3	CWEN2	CWEN1	CWEN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWENR0 field descriptions

Field	Description
0 CWEN31	Watchdog enable for channel 31. When set, Watchdog feature is enabled for channel 31.
1 CWEN30	Watchdog enable for channel 30. When set, Watchdog feature is enabled for channel 30.
2 CWEN29	Watchdog enable for channel 29. When set, Watchdog feature is enabled for channel 29.
3 CWEN28	Watchdog enable for channel 28. When set, Watchdog feature is enabled for channel 28.
4 CWEN27	Watchdog enable for channel 27. When set, Watchdog feature is enabled for channel 27.
5 CWEN26	Watchdog enable for channel 26. When set, Watchdog feature is enabled for channel 26.
6 CWEN25	Watchdog enable for channel 25. When set, Watchdog feature is enabled for channel 25.
7 CWEN24	Watchdog enable for channel 24. When set, Watchdog feature is enabled for channel 24.
8 CWEN23	Watchdog enable for channel 23. When set, Watchdog feature is enabled for channel 23.

Table continues on the next page...

ADC_CWENR0 field descriptions (continued)

Field	Description
9 CWEN22	Watchdog enable for channel 22. When set, Watchdog feature is enabled for channel 22.
10 CWEN21	Watchdog enable for channel 21. When set, Watchdog feature is enabled for channel 21.
11 CWEN20	Watchdog enable for channel 20. When set, Watchdog feature is enabled for channel 20.
12 CWEN19	Watchdog enable for channel 19. When set, Watchdog feature is enabled for channel 19.
13 CWEN18	Watchdog enable for channel 18. When set, Watchdog feature is enabled for channel 18.
14 CWEN17	Watchdog enable for channel 17. When set, Watchdog feature is enabled for channel 17.
15 CWEN16	Watchdog enable for channel 16. When set, Watchdog feature is enabled for channel 16.
16 CWEN15	Watchdog enable for channel 15. When set, Watchdog feature is enabled for channel 15.
17 CWEN14	Watchdog enable for channel 14. When set, Watchdog feature is enabled for channel 14.
18 CWEN13	Watchdog enable for channel 13. When set, Watchdog feature is enabled for channel 13.
19 CWEN12	Watchdog enable for channel 12. When set, Watchdog feature is enabled for channel 12.
20 CWEN11	Watchdog enable for channel 11. When set, Watchdog feature is enabled for channel 11.
21 CWEN10	Watchdog enable for channel 10. When set, Watchdog feature is enabled for channel 10.
22 CWEN9	Watchdog enable for channel 9. When set, Watchdog feature is enabled for channel 9.
23 CWEN8	Watchdog enable for channel 8. When set, Watchdog feature is enabled for channel 8.
24 CWEN7	Watchdog enable for channel 7. When set, Watchdog feature is enabled for channel 7.
25 CWEN6	Watchdog enable for channel 6. When set, Watchdog feature is enabled for channel 6.
26 CWEN5	Watchdog enable for channel 5.

Table continues on the next page...

ADC_CWENR0 field descriptions (continued)

Field	Description
	When set, Watchdog feature is enabled for channel 5.
27 CWEN4	Watchdog enable for channel 4. When set, Watchdog feature is enabled for channel 4.
28 CWEN3	Watchdog enable for channel 3. When set, Watchdog feature is enabled for channel 3.
29 CWEN2	Watchdog enable for channel 2. When set, Watchdog feature is enabled for channel 2.
30 CWEN1	Watchdog enable for channel 1. When set, Watchdog feature is enabled for channel 1.
31 CWEN0	Watchdog enable for channel 0. When set, Watchdog feature is enabled for channel 0.

34.5.49 Channel Watchdog Enable Register 1 (ADC_CWENR1)

Controls whether the watchdog is enabled for channels 32 to 63.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2E4h offset = 2E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CWEN63	CWEN62	CWEN61	CWEN60	CWEN59	CWEN58	CWEN57	CWEN56	CWEN55	CWEN54	CWEN53	CWEN52	CWEN51	CWEN50	CWEN49	CWEN48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CWEN47	CWEN46	CWEN45	CWEN44	CWEN43	CWEN42	CWEN41	CWEN40	CWEN39	CWEN38	CWEN37	CWEN36	CWEN35	CWEN34	CWEN33	CWEN32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWENR1 field descriptions

Field	Description
0 CWEN63	Watchdog enable for channel 63. When set, Watchdog feature is enabled for channel 63.
1 CWEN62	Watchdog enable for channel 62. When set, Watchdog feature is enabled for channel 62.
2 CWEN61	Watchdog enable for channel 61. When set, Watchdog feature is enabled for channel 61.
3 CWEN60	Watchdog enable for channel 60. When set, Watchdog feature is enabled for channel 60.
4 CWEN59	Watchdog enable for channel 59. When set, Watchdog feature is enabled for channel 59.
5 CWEN58	Watchdog enable for channel 58. When set, Watchdog feature is enabled for channel 58.
6 CWEN57	Watchdog enable for channel 57. When set, Watchdog feature is enabled for channel 57.
7 CWEN56	Watchdog enable for channel 56. When set, Watchdog feature is enabled for channel 56.
8 CWEN55	Watchdog enable for channel 55. When set, Watchdog feature is enabled for channel 55.
9 CWEN54	Watchdog enable for channel 54. When set, Watchdog feature is enabled for channel 54.
10 CWEN53	Watchdog enable for channel 53. When set, Watchdog feature is enabled for channel 53.
11 CWEN52	Watchdog enable for channel 52. When set, Watchdog feature is enabled for channel 52.
12 CWEN51	Watchdog enable for channel 51. When set, Watchdog feature is enabled for channel 51.
13 CWEN50	Watchdog enable for channel 50. When set, Watchdog feature is enabled for channel 50.
14 CWEN49	Watchdog enable for channel 49. When set, Watchdog feature is enabled for channel 49.
15 CWEN48	Watchdog enable for channel 48. When set, Watchdog feature is enabled for channel 48.
16 CWEN47	Watchdog enable for channel 47. When set, Watchdog feature is enabled for channel 47.
17 CWEN46	Watchdog enable for channel 46.

Table continues on the next page...

ADC_CWENR1 field descriptions (continued)

Field	Description
	When set, Watchdog feature is enabled for channel 46.
18 CWEN45	Watchdog enable for channel 45. When set, Watchdog feature is enabled for channel 45.
19 CWEN44	Watchdog enable for channel 44. When set, Watchdog feature is enabled for channel 44.
20 CWEN43	Watchdog enable for channel 43. When set, Watchdog feature is enabled for channel 43.
21 CWEN42	Watchdog enable for channel 42. When set, Watchdog feature is enabled for channel 42.
22 CWEN41	Watchdog enable for channel 41. When set, Watchdog feature is enabled for channel 41.
23 CWEN40	Watchdog enable for channel 40. When set, Watchdog feature is enabled for channel 40.
24 CWEN39	Watchdog enable for channel 39. When set, Watchdog feature is enabled for channel 39.
25 CWEN38	Watchdog enable for channel 38. When set, Watchdog feature is enabled for channel 38.
26 CWEN37	Watchdog enable for channel 37. When set, Watchdog feature is enabled for channel 37.
27 CWEN36	Watchdog enable for channel 36. When set, Watchdog feature is enabled for channel 36.
28 CWEN35	Watchdog enable for channel 35. When set, Watchdog feature is enabled for channel 35.
29 CWEN34	Watchdog enable for channel 34. When set, Watchdog feature is enabled for channel 34.
30 CWEN33	Watchdog enable for channel 33. When set, Watchdog feature is enabled for channel 33.
31 CWEN32	Watchdog enable for channel 32. When set, Watchdog feature is enabled for channel 32.

34.5.50 Channel Watchdog Enable Register 2 (ADC_CWENR2)

Controls whether the watchdog is enabled for channels 64 to 95.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2E8h offset = 2E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CWEN95	CWEN94	CWEN93	CWEN92	CWEN91	CWEN90	CWEN89	CWEN88	CWEN87	CWEN86	CWEN85	CWEN84	CWEN83	CWEN82	CWEN81	CWEN80
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CWEN79	CWEN78	CWEN77	CWEN76	CWEN75	CWEN74	CWEN73	CWEN72	CWEN71	CWEN70	CWEN69	CWEN68	CWEN67	CWEN66	CWEN65	CWEN64
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_CWENR2 field descriptions

Field	Description
0 CWEN95	Watchdog enable for channel 95. When set, Watchdog feature is enabled for channel 95.
1 CWEN94	Watchdog enable for channel 94. When set, Watchdog feature is enabled for channel 94.
2 CWEN93	Watchdog enable for channel 93. When set, Watchdog feature is enabled for channel 93.
3 CWEN92	Watchdog enable for channel 92. When set, Watchdog feature is enabled for channel 92.
4 CWEN91	Watchdog enable for channel 91. When set, Watchdog feature is enabled for channel 91.
5 CWEN90	Watchdog enable for channel 90. When set, Watchdog feature is enabled for channel 90.
6 CWEN89	Watchdog enable for channel 89. When set, Watchdog feature is enabled for channel 89.
7 CWEN88	Watchdog enable for channel 88. When set, Watchdog feature is enabled for channel 88.
8 CWEN87	Watchdog enable for channel 87. When set, Watchdog feature is enabled for channel 87.

Table continues on the next page...

ADC_CWENR2 field descriptions (continued)

Field	Description
9 CWEN86	Watchdog enable for channel 86. When set, Watchdog feature is enabled for channel 86.
10 CWEN85	Watchdog enable for channel 85. When set, Watchdog feature is enabled for channel 85.
11 CWEN84	Watchdog enable for channel 84. When set, Watchdog feature is enabled for channel 84.
12 CWEN83	Watchdog enable for channel 83. When set, Watchdog feature is enabled for channel 83.
13 CWEN82	Watchdog enable for channel 82. When set, Watchdog feature is enabled for channel 82.
14 CWEN81	Watchdog enable for channel 81. When set, Watchdog feature is enabled for channel 81.
15 CWEN80	Watchdog enable for channel 80. When set, Watchdog feature is enabled for channel 80.
16 CWEN79	Watchdog enable for channel 79. When set, Watchdog feature is enabled for channel 79.
17 CWEN78	Watchdog enable for channel 78. When set, Watchdog feature is enabled for channel 78.
18 CWEN77	Watchdog enable for channel 77. When set, Watchdog feature is enabled for channel 77.
19 CWEN76	Watchdog enable for channel 76. When set, Watchdog feature is enabled for channel 76.
20 CWEN75	Watchdog enable for channel 75. When set, Watchdog feature is enabled for channel 75.
21 CWEN74	Watchdog enable for channel 74. When set, Watchdog feature is enabled for channel 74.
22 CWEN73	Watchdog enable for channel 73. When set, Watchdog feature is enabled for channel 73.
23 CWEN72	Watchdog enable for channel 72. When set, Watchdog feature is enabled for channel 72.
24 CWEN71	Watchdog enable for channel 71. When set, Watchdog feature is enabled for channel 71.
25 CWEN70	Watchdog enable for channel 70. When set, Watchdog feature is enabled for channel 70.
26 CWEN69	Watchdog enable for channel 69.

Table continues on the next page...

ADC_CWENR2 field descriptions (continued)

Field	Description
	When set, Watchdog feature is enabled for channel 69.
27 CWEN68	Watchdog enable for channel 68. When set, Watchdog feature is enabled for channel 68.
28 CWEN67	Watchdog enable for channel 67. When set, Watchdog feature is enabled for channel 67.
29 CWEN66	Watchdog enable for channel 66. When set, Watchdog feature is enabled for channel 66.
30 CWEN65	Watchdog enable for channel 65. When set, Watchdog feature is enabled for channel 65.
31 CWEN64	Watchdog enable for channel 64. When set, Watchdog feature is enabled for channel 64.

34.5.51 Analog Watchdog Out of Range Register 0 (ADC_AWORR0)

Analog watchdog out of range register for channels 0 to 31.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2F0h offset = 2F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH31	AWOR_CH30	AWOR_CH29	AWOR_CH28	AWOR_CH27	AWOR_CH26	AWOR_CH25	AWOR_CH24	AWOR_CH23	AWOR_CH22	AWOR_CH21	AWOR_CH20	AWOR_CH19	AWOR_CH18	AWOR_CH17	AWOR_CH16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH15	AWOR_CH14	AWOR_CH13	AWOR_CH12	AWOR_CH11	AWOR_CH10	AWOR_CH9	AWOR_CH8	AWOR_CH7	AWOR_CH6	AWOR_CH5	AWOR_CH4	AWOR_CH3	AWOR_CH2	AWOR_CH1	AWOR_CH0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_AWORR0 field descriptions

Field	Description
0 AWOR_CH31	Analog watchdog out of range for channel 31. When set, indicates the converted data is out of range for channel 31.
1 AWOR_CH30	Analog watchdog out of range for channel 30. When set, indicates the converted data is out of range for channel 30.
2 AWOR_CH29	Analog watchdog out of range for channel 29. When set, indicates the converted data is out of range for channel 29.
3 AWOR_CH28	Analog watchdog out of range for channel 28. When set, indicates the converted data is out of range for channel 28.
4 AWOR_CH27	Analog watchdog out of range for channel 27. When set, indicates the converted data is out of range for channel 27.
5 AWOR_CH26	Analog watchdog out of range for channel 26. When set, indicates the converted data is out of range for channel 26.
6 AWOR_CH25	Analog watchdog out of range for channel 25. When set, indicates the converted data is out of range for channel 25.
7 AWOR_CH24	Analog watchdog out of range for channel 24. When set, indicates the converted data is out of range for channel 24.
8 AWOR_CH23	Analog watchdog out of range for channel 23. When set, indicates the converted data is out of range for channel 23.
9 AWOR_CH22	Analog watchdog out of range for channel 22. When set, indicates the converted data is out of range for channel 22.
10 AWOR_CH21	Analog watchdog out of range for channel 21. When set, indicates the converted data is out of range for channel 21.
11 AWOR_CH20	Analog watchdog out of range for channel 20. When set, indicates the converted data is out of range for channel 20.
12 AWOR_CH19	Analog watchdog out of range for channel 19.

Table continues on the next page...

ADC_AWORR0 field descriptions (continued)

Field	Description
	When set, indicates the converted data is out of range for channel 19.
13 AWOR_CH18	Analog watchdog out of range for channel 18. When set, indicates the converted data is out of range for channel 18.
14 AWOR_CH17	Analog watchdog out of range for channel 17. When set, indicates the converted data is out of range for channel 17.
15 AWOR_CH16	Analog watchdog out of range for channel 16. When set, indicates the converted data is out of range for channel 16.
16 AWOR_CH15	Analog watchdog out of range for channel 15. When set, indicates the converted data is out of range for channel 15.
17 AWOR_CH14	Analog watchdog out of range for channel 14. When set, indicates the converted data is out of range for channel 14.
18 AWOR_CH13	Analog watchdog out of range for channel 13. When set, indicates the converted data is out of range for channel 13.
19 AWOR_CH12	Analog watchdog out of range for channel 12. When set, indicates the converted data is out of range for channel 12.
20 AWOR_CH11	Analog watchdog out of range for channel 11. When set, indicates the converted data is out of range for channel 11.
21 AWOR_CH10	Analog watchdog out of range for channel 10. When set, indicates the converted data is out of range for channel 10.
22 AWOR_CH9	Analog watchdog out of range for channel 9. When set, indicates the converted data is out of range for channel 9.
23 AWOR_CH8	Analog watchdog out of range for channel 8. When set, indicates the converted data is out of range for channel 8.
24 AWOR_CH7	Analog watchdog out of range for channel 7. When set, indicates the converted data is out of range for channel 7.
25 AWOR_CH6	Analog watchdog out of range for channel 6. When set, indicates the converted data is out of range for channel 6.
26 AWOR_CH5	Analog watchdog out of range for channel 5. When set, indicates the converted data is out of range for channel 5.
27 AWOR_CH4	Analog watchdog out of range for channel 4. When set, indicates the converted data is out of range for channel 4.
28 AWOR_CH3	Analog watchdog out of range for channel 3. When set, indicates the converted data is out of range for channel 3.
29 AWOR_CH2	Analog watchdog out of range for channel 2. When set, indicates the converted data is out of range for channel 2.

Table continues on the next page...

ADC_AWORR0 field descriptions (continued)

Field	Description
30 AWOR_CH1	Analog watchdog out of range for channel 1. When set, indicates the converted data is out of range for channel 1.
31 AWOR_CH0	Analog watchdog out of range for channel 0. When set, indicates the converted data is out of range for channel 0.

34.5.52 Analog Watchdog Out of Range Register 1 (ADC_AWORR1)

Analog watchdog out of range register for channels 32 to 63.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2F4h offset = 2F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH63	AWOR_CH62	AWOR_CH61	AWOR_CH60	AWOR_CH59	AWOR_CH58	AWOR_CH57	AWOR_CH56	AWOR_CH55	AWOR_CH54	AWOR_CH53	AWOR_CH52	AWOR_CH51	AWOR_CH50	AWOR_CH49	AWOR_CH48
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH47	AWOR_CH46	AWOR_CH45	AWOR_CH44	AWOR_CH43	AWOR_CH42	AWOR_CH41	AWOR_CH40	AWOR_CH39	AWOR_CH38	AWOR_CH37	AWOR_CH36	AWOR_CH35	AWOR_CH34	AWOR_CH33	AWOR_CH32
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_AWORR1 field descriptions

Field	Description
0 AWOR_CH63	Analog watchdog out of range for channel 63. When set, indicates the converted data is out of range for channel 63.
1 AWOR_CH62	Analog watchdog out of range for channel 62. When set, indicates the converted data is out of range for channel 62.
2 AWOR_CH61	Analog watchdog out of range for channel 61. When set, indicates the converted data is out of range for channel 61.
3 AWOR_CH60	Analog watchdog out of range for channel 60. When set, indicates the converted data is out of range for channel 60.
4 AWOR_CH59	Analog watchdog out of range for channel 59. When set, indicates the converted data is out of range for channel 59.
5 AWOR_CH58	Analog watchdog out of range for channel 58. When set, indicates the converted data is out of range for channel 58.
6 AWOR_CH57	Analog watchdog out of range for channel 57. When set, indicates the converted data is out of range for channel 57.
7 AWOR_CH56	Analog watchdog out of range for channel 56. When set, indicates the converted data is out of range for channel 56.
8 AWOR_CH55	Analog watchdog out of range for channel 55. When set, indicates the converted data is out of range for channel 55.
9 AWOR_CH54	Analog watchdog out of range for channel 54. When set, indicates the converted data is out of range for channel 54.
10 AWOR_CH53	Analog watchdog out of range for channel 53. When set, indicates the converted data is out of range for channel 53.
11 AWOR_CH52	Analog watchdog out of range for channel 52. When set, indicates the converted data is out of range for channel 52.
12 AWOR_CH51	Analog watchdog out of range for channel 51. When set, indicates the converted data is out of range for channel 51.
13 AWOR_CH50	Analog watchdog out of range for channel 50. When set, indicates the converted data is out of range for channel 50.
14 AWOR_CH49	Analog watchdog out of range for channel 49. When set, indicates the converted data is out of range for channel 49.
15 AWOR_CH48	Analog watchdog out of range for channel 48. When set, indicates the converted data is out of range for channel 48.
16 AWOR_CH47	Analog watchdog out of range for channel 47. When set, indicates the converted data is out of range for channel 47.
17 AWOR_CH46	Analog watchdog out of range for channel 46.

Table continues on the next page...

ADC_AWORR1 field descriptions (continued)

Field	Description
	When set, indicates the converted data is out of range for channel 46.
18 AWOR_CH45	Analog watchdog out of range for channel 45. When set, indicates the converted data is out of range for channel 45.
19 AWOR_CH44	Analog watchdog out of range for channel 44. When set, indicates the converted data is out of range for channel 44.
20 AWOR_CH43	Analog watchdog out of range for channel 43. When set, indicates the converted data is out of range for channel 43.
21 AWOR_CH42	Analog watchdog out of range for channel 42. When set, indicates the converted data is out of range for channel 42.
22 AWOR_CH41	Analog watchdog out of range for channel 41. When set, indicates the converted data is out of range for channel 41.
23 AWOR_CH40	Analog watchdog out of range for channel 40. When set, indicates the converted data is out of range for channel 40.
24 AWOR_CH39	Analog watchdog out of range for channel 39. When set, indicates the converted data is out of range for channel 39.
25 AWOR_CH38	Analog watchdog out of range for channel 38. When set, indicates the converted data is out of range for channel 38.
26 AWOR_CH37	Analog watchdog out of range for channel 37. When set, indicates the converted data is out of range for channel 37.
27 AWOR_CH36	Analog watchdog out of range for channel 36. When set, indicates the converted data is out of range for channel 36.
28 AWOR_CH35	Analog watchdog out of range for channel 35. When set, indicates the converted data is out of range for channel 35.
29 AWOR_CH34	Analog watchdog out of range for channel 34. When set, indicates the converted data is out of range for channel 34.
30 AWOR_CH33	Analog watchdog out of range for channel 33. When set, indicates the converted data is out of range for channel 33.
31 AWOR_CH32	Analog watchdog out of range for channel 32. When set, indicates the converted data is out of range for channel 32.

34.5.53 Analog Watchdog Out of Range Register 2 (ADC_AWORR2)

Analog watchdog out of range register for channels 64 to 95.

NOTE

This register may contain fields for channels that have not been implemented. See the chip-specific information to find which channels are applicable.

Address: 0h base + 2F8h offset = 2F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	AWOR_CH95	AWOR_CH94	AWOR_CH93	AWOR_CH92	AWOR_CH91	AWOR_CH90	AWOR_CH89	AWOR_CH88	AWOR_CH87	AWOR_CH86	AWOR_CH85	AWOR_CH84	AWOR_CH83	AWOR_CH82	AWOR_CH81	AWOR_CH80
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	AWOR_CH79	AWOR_CH78	AWOR_CH77	AWOR_CH76	AWOR_CH75	AWOR_CH74	AWOR_CH73	AWOR_CH72	AWOR_CH71	AWOR_CH70	AWOR_CH69	AWOR_CH68	AWOR_CH67	AWOR_CH66	AWOR_CH65	AWOR_CH64
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADC_AWORR2 field descriptions

Field	Description
0 AWOR_CH95	Analog watchdog out of range for channel 95. When set, indicates the converted data is out of range for channel 95.
1 AWOR_CH94	Analog watchdog out of range for channel 94. When set, indicates the converted data is out of range for channel 94.
2 AWOR_CH93	Analog watchdog out of range for channel 93. When set, indicates the converted data is out of range for channel 93.
3 AWOR_CH92	Analog watchdog out of range for channel 92. When set, indicates the converted data is out of range for channel 92.
4 AWOR_CH91	Analog watchdog out of range for channel 91. When set, indicates the converted data is out of range for channel 91.

Table continues on the next page...

ADC_AWORR2 field descriptions (continued)

Field	Description
5 AWOR_CH90	Analog watchdog out of range for channel 90. When set, indicates the converted data is out of range for channel 90.
6 AWOR_CH89	Analog watchdog out of range for channel 89. When set, indicates the converted data is out of range for channel 89.
7 AWOR_CH88	Analog watchdog out of range for channel 88. When set, indicates the converted data is out of range for channel 88.
8 AWOR_CH87	Analog watchdog out of range for channel 87. When set, indicates the converted data is out of range for channel 87.
9 AWOR_CH86	Analog watchdog out of range for channel 86. When set, indicates the converted data is out of range for channel 86.
10 AWOR_CH85	Analog watchdog out of range for channel 85. When set, indicates the converted data is out of range for channel 85.
11 AWOR_CH84	Analog watchdog out of range for channel 84. When set, indicates the converted data is out of range for channel 84.
12 AWOR_CH83	Analog watchdog out of range for channel 83. When set, indicates the converted data is out of range for channel 83.
13 AWOR_CH82	Analog watchdog out of range for channel 82. When set, indicates the converted data is out of range for channel 82.
14 AWOR_CH81	Analog watchdog out of range for channel 81. When set, indicates the converted data is out of range for channel 81.
15 AWOR_CH80	Analog watchdog out of range for channel 80. When set, indicates the converted data is out of range for channel 80.
16 AWOR_CH79	Analog watchdog out of range for channel 79. When set, indicates the converted data is out of range for channel 79.
17 AWOR_CH78	Analog watchdog out of range for channel 78. When set, indicates the converted data is out of range for channel 78.
18 AWOR_CH77	Analog watchdog out of range for channel 77. When set, indicates the converted data is out of range for channel 77.
19 AWOR_CH76	Analog watchdog out of range for channel 76. When set, indicates the converted data is out of range for channel 76.
20 AWOR_CH75	Analog watchdog out of range for channel 75. When set, indicates the converted data is out of range for channel 75.
21 AWOR_CH74	Analog watchdog out of range for channel 74. When set, indicates the converted data is out of range for channel 74.
22 AWOR_CH73	Analog watchdog out of range for channel 73.

Table continues on the next page...

ADC_AWORR2 field descriptions (continued)

Field	Description
	When set, indicates the converted data is out of range for channel 73.
23 AWOR_CH72	Analog watchdog out of range for channel 72. When set, indicates the converted data is out of range for channel 72.
24 AWOR_CH71	Analog watchdog out of range for channel 71. When set, indicates the converted data is out of range for channel 71.
25 AWOR_CH70	Analog watchdog out of range for channel 70. When set, indicates the converted data is out of range for channel 70.
26 AWOR_CH69	Analog watchdog out of range for channel 69. When set, indicates the converted data is out of range for channel 69.
27 AWOR_CH68	Analog watchdog out of range for channel 68. When set, indicates the converted data is out of range for channel 68.
28 AWOR_CH67	Analog watchdog out of range for channel 67. When set, indicates the converted data is out of range for channel 67.
29 AWOR_CH66	Analog watchdog out of range for channel 66. When set, indicates the converted data is out of range for channel 66.
30 AWOR_CH65	Analog watchdog out of range for channel 65. When set, indicates the converted data is out of range for channel 65.
31 AWOR_CH64	Analog watchdog out of range for channel 64. When set, indicates the converted data is out of range for channel 64.

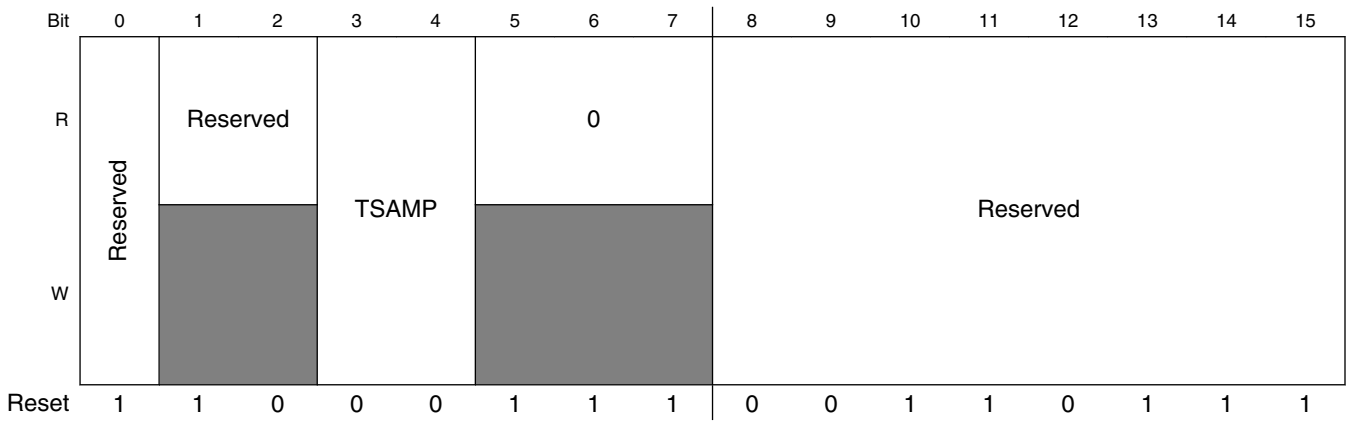
34.5.54 Calibration, BIST Control and status Register (ADC_CALBISTREG)

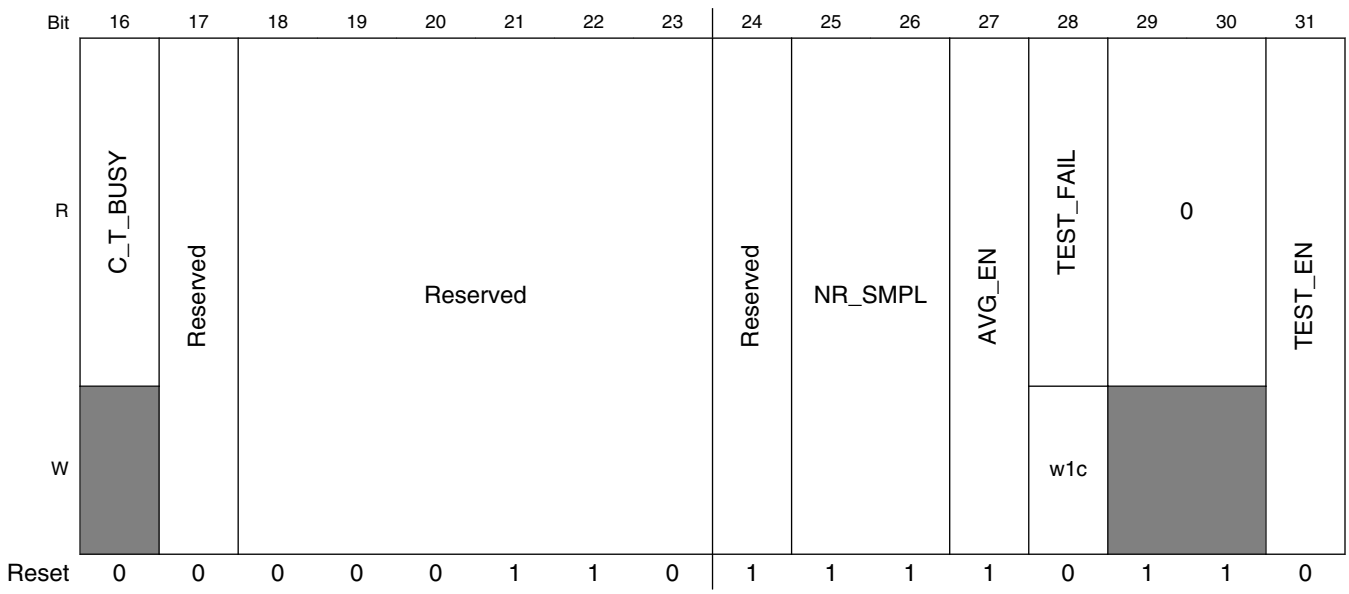
This register controls the Offset calculation, Calibration and BIST functionality with Averaging option. The averaging function has no effect in other conversion modes.

NOTE

The reset value shown for this register is for a 12-bit resolution ADC. The 10-bit ADC reset value is 4137_06F6h.

Address: 0h base + 3A0h offset = 3A0h





ADC_CALBISTREG field descriptions

Field	Description
0 Reserved	This field is reserved. When writing to this register, do not change the default value of this field.
1–2 Reserved	This field is reserved. When writing to this register, do not change the default value of this field. **NOTE:** The reset value of this field depends on the resolution of the ADC instance. • Reset value is 01b for 12-bit ADC instances • Reset value is 10b for 10-bit ADC instances
3–4 TSAMP	Test Sample period in Calibration, BIST and Offset calculation process. 00 22 cycles of ADC clk 01 8 cycles of ADC clk 10 16 cycle of ADC clk 11 32 cycle of ADC clk
5–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ADC_CALBISTREG field descriptions (continued)

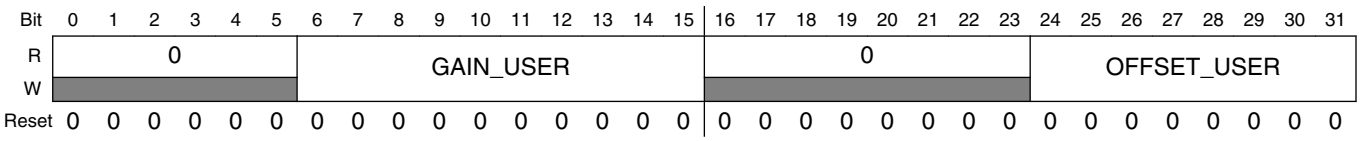
Field	Description
8–15 Reserved	This field is reserved. When writing to this register, do not change the default value of this field.
16 C_T_BUSY	This bit indicates that ADC is doing internal operations for BIST or Calibration. This is set when the TEST_EN bit is set. 0 ADC is ready for use 1 ADC is busy in calibration or test operation
17 Reserved	This field is reserved.
18–23 Reserved	This field is reserved. When writing to this register, do not change the default value of this field.
24 Reserved	This field is reserved. The default value must not change. Always write the default value to this field.
25–26 NR_SMPL	Number of Samples for averaging. **NOTE:** The sample setting is providing a trade-off between calibration time and accuracy. For best accuracy 512 samples are recommended. The 16 sample setting shall only be used for applications which require very fast calibration time at the expense of accuracy. 00 16 samples 01 32 samples 10 128 samples 11 512 samples
27 AVG_EN	Average Enable (for Calibration only). 0 Disable 1 Enable
28 TEST_FAIL	Test Fail. This bit indicates the status of Calibration output. This bit sets automatically if tests are aborted prematurely by any normal conversion or tests complete but one of them fails. This bit is cleared by writing 1 to it or on restarting of test by writing 1 to TEST_EN 0 Test passed 1 Test failed
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TEST_EN	Enable the test. 0 Disable the test 1 Enable the test (self clearing) (once set it cannot be overwritten till it resets)

34.5.55 Offset and Gain User Register (ADC_OFSGNUSR)

This register contains the user configured offset and Gain values used by ADC for data processing and have all the bits valid (MSB through LSB).

OFFSET and Gain Calibration values can be either positive or negative, so they need to be in 2's complement format with MSB as sign bit.

Address: 0h base + 3A8h offset = 3A8h



ADC_OFSGNUSR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 GAIN_USER	Gain User (2's complement format)
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 OFFSET_USER	Offset User (2's complement format)

34.6 Functional description

The following sections detail the ADCD interfaces as well as the functionality of this module.

34.6.1 Conversion

After power-up or reset, the ADC is in power-down mode until the MCR[PWDN] field is written. There are some configurations, listed below, available only in power-down mode, that must be handled before exiting power down.

- Clock selection for SAR Controller (AD_clk): MCR [ADCLKSEL]
- Operating mode of ADC for BCTU (Trigger / Control): MCR [CTU_MODE] (if available)

After the above configurations, ADC need to be taken out from power-down mode by writing '0' into MCR [PWDN], Conversions can be initiated now in the following modes:

- Normal
- Injected
- BCTU Triggered

34.6.1.1 Normal conversion

The conversion mode generally used to convert channels is called normal conversion. Programming the available Normal Conversion Mask Register(s) (ADC_NCMR*n*) selects the channel(s). Each channel can be individually enabled by setting a bit in the corresponding field of the appropriate NCMR*n* register.

- NCMR0 serves *precision* channels:
  - 0 - 23 (ADC0—10 bit)
  - 0 - 15 (ADC1—12 bit)
- NCMR1 serves *internal* channels:
  - 32 - 63 (ADC0—10 bit)
  - 32 - 57 (ADC1—12 bit)
- NCMR2 serves *external* channels:
  - 64 to 95 (ADC0—10 bit)
  - 64 to 95 (ADC1—12 bit)

The mask register(s) must be programmed before starting the conversion and cannot be changed until the conversion of all the selected channels ends. Conversion always starts from the lowest channel number selected and sequentially goes to the highest number.

34.6.1.1.1 Starting a normal conversion

After programming the MCR and NCMR0*n* register(s), a normal conversion is started via *software* by setting the MCR[NSTART] bit.

34.6.1.2 Normal conversion operating modes

Two operating modes are available for normal conversion:

- One-Shot mode
- Scan mode

The MCR[MODE] bit determines which mode is used during a normal conversion. The first phase of the conversion process involves sampling the analog channel. The next phase is the conversion phase, during which the sampled analog value is converted to digital as shown in the figure below.

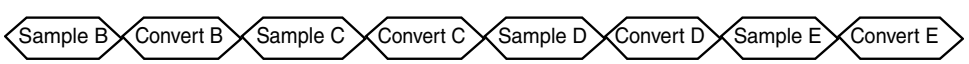


Figure 34-3. Normal conversion flow

34.6.1.2.1 One shot mode

In one shot mode (MCR[MODE] = 0) a sequential conversion specified in the following mask register(s) is performed only once:

- NCMR0
- NCMR1
- NCMR2

At the end of each conversion, the digital result of the conversion is stored into the corresponding data register(CDRx, x = channel number).

For example: Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in One-Shot mode. Conversion starts with channel B, followed by the conversion of channels D and E. At the end of conversion of channel E, the scanning of channels will stop.

The MSR[NSTART] status bit is automatically set when the normal conversion starts. At the same time, MCR[NSTART]is reset by hardware, allowing the software to program a new start of conversion in advance. In this case, the newly requested conversion starts after completing the running/current conversion. However, for the sake of simplicity and straightforwardness, it is advisable to program the NSTART bit for the new conversion only after the completion of the running conversion. To know the end of conversion of running chain, application will either wait for an End of Chain (ECH) Interrupt or check the status of MSR[NSTART] prior to setting MCR[NSTART] again.

34.6.1.2.2 Scan mode

In scan mode (MCR[MODE] = 1), a sequential conversion of *n* channels specified in the NCMR*n* register(s) is continuously performed. At the end of each conversion, the converted result is stored into the corresponding data registers, as in One-Shot mode.

The MSR[NSTART] status bit is automatically set when the normal conversion starts. Unlike One-Shot mode, the MCR[NSTART] bit is not reset automatically in Scan mode. It can be reset by software when you need to stop Scan mode. In that case, the ADC completes the current chain and, after the last conversion, resets MSR[NSTART].

For example: Channels A-B-C-D-E-F-G-H are present in the device where channels B-D-E are to be converted in Scan mode. MCR[MODE] = 1 is set for Scan mode. Conversion starts from channel B, followed by the conversion of channels D and E. At the end of the conversion of channel E, the conversion of channel B starts again, followed by the conversion of channels D and E. This sequence repeats itself until MCR[NSTART] is reset by software.

This scan chain can be stopped by writing 0 to MCR[NSTART]. The conversion stops when the ongoing chain is finished.

34.6.1.2.3 End of conversion

In both modes, at the end of each conversion an end-of-conversion (EOC) interrupt is issued if enabled by the corresponding mask bit in the following register(s):

- CIMR0
- CIMR1
- CIMR2

Additionally, the EOC interrupt must enabled in the IMR register. After the conversion of all the channels selected in the NCMR*n* register(s) is completed and the conversion operation is considered finished, ISR[ECH] is set and an end-of-chain (ECH) interrupt is issued (if enabled in IMR[MSKECH]). The corresponding channel bit within the appropriate CEOCFR*n* register is updated to indicate that data is available on the data register (CDR*n*) of the respective channel. If there is no channel selected in the NCMR*n* register(s) and there is a start of conversion trigger, then ISR[ECH] is set and an ECH interrupt is immediately issued (if enabled).

NOTE

To add or remove channels to/from the ongoing chain:

1. Clear the MCR[NSTART] bit
2. Set/clear mask bit for the needed channels in NCMR*n*
3. Set the MCR[NSTART] bit

34.6.1.3 Injected conversion

An injected conversion is a conversion chain that can be injected at any time, including into an ongoing normal conversion chain. To do so, configure the injected mask register(s):

- JCMR0
- JCMR1
- JCMR2

As in a normal conversion, each internal or external channel can be individually selected for injected conversion.

- JCMR0 controls *precision* channels:
  - 0 - 23 (ADC0—10 bit)
  - 0 - 15 (ADC1—12 bit)

- JCMR1 controls *internal* channels:
  - 32 - 63 (ADC0—10 bit)
  - 32 - 57 (ADC1—12 bit)
- JCMR2 controls *external* channels:
  - 64 to 95 (ADC0—10 bit)
  - 64 to 95 (ADC1—12 bit)

Injected conversion can only be run in One-Shot mode, and can only interrupt a normal conversion. Please note, that a BCTU conversion cannot be interrupted by an injected conversion. In this case, the injected conversion is discarded. When an injected chain is inserted into an ongoing normal chain, the ongoing normal channel is suspended and the injected channel request is processed. After completing the last channel in the injected chain, normal conversion resumes from the channel at which it was suspended as shown in the following figure.

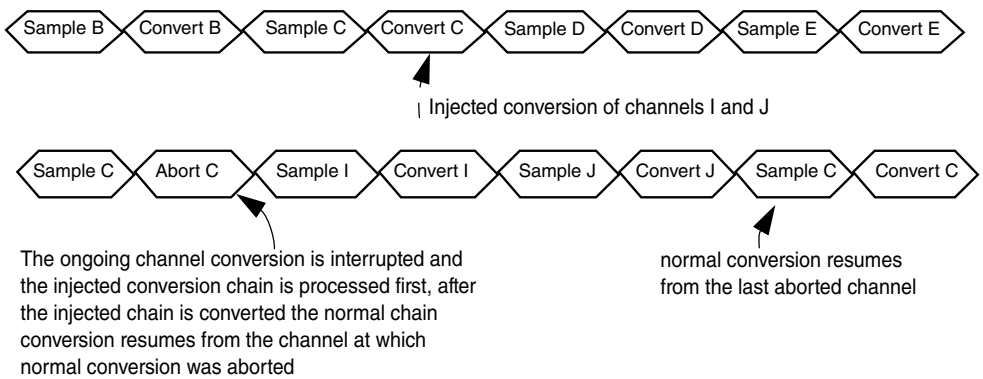


Figure 34-4. Injected sample/conversion sequence

34.6.1.3.1 Starting injected conversion

The injected conversion can be started via *software* by setting the MCR[JSTART] bit to start an injected conversion; the current normal conversion is suspended and the injected chain is converted.

To start an injected conversion via *external trigger*, enable external triggering by setting MCR[JTRGEN]=1. A programmed event (rising/falling edge depending on MCR[JEDGE] bit) on the injection trigger input starts the injected conversion.

The MSR[JSTART] status bit is automatically set when the injected conversion starts. At the same time, the MCR[JSTART] bit is reset, allowing software to program a new start of conversion in advance. In that case, the new requested conversion starts after the running conversion is completed.

At the end of each conversion, an End Of Injected Conversion (JEOC) interrupt is issued (if enabled by the corresponding mask bit in IMR) and at the end of the chain an End Of Injected Chain (JECH) interrupt is issued (if enabled by the corresponding mask bit in IMR). The status bit ISR[JECH] is also set.

If the content of all the injected conversion mask registers (JCMRn) is zero (i.e., no channel is selected) the interrupt JECH is immediately issued to resume normal operation or conversion mode.

The corresponding channel bit in the appropriate CEOCFR*n* register is updated to indicate that data is available in the data register (CDR) of the respective channel.

Once started, injected chain conversion cannot be interrupted by Normal or another Injected conversion. A BCTU trigger has the higher priority; injected conversion has a lower priority. Therefore, only a BCTU trigger can interrupt an ongoing injected conversion. In this case, the injected conversion will be discarded.

34.6.1.4 Aborting conversions

Two different abort functions are provided:

- A single channel abort
- A chain abort

You can abort the ongoing conversion by setting the MCR[ABORT] bit. The current conversion is aborted and the conversion of the next channel of the chain is immediately started. The abort action may take 1 to 4 cycles of the bus clock to abort the current conversion depending on the state of the conversion. At the start of any conversion and at the end of a particular conversion, when internal state counters are changing, then abort is delayed by a maximum of 3 cycles to take all states in stable condition. Abort also should not be programmed along with and within 3 cycles of programming the MCR[NSTART] or MCR[JSTART]. This may not have a significant impact from software point of view since on any SoC the next write access comes only after a minimum of 3 clock cycles.

NOTE

Programming Abort or Abort chain with NSTART or JSTART in idle condition has no effect.

For a Normal conversion chain having only one channel:

1. Abort re-runs the conversion again from the Abort point
2. If Interrupted by BCTU (in Trigger mode) Normal conversion runs twice instead of just once

In the case of an abort operation, the MSR[NSTART/JSTART] bit remains set, if not in last channel of chain, and the MCR[ABORT] bit is reset as soon as the channel is aborted. The EOC corresponding to the aborted channel is not generated. This behavior is true for Normal or Injected conversion modes. If the last channel of a chain is aborted, the end of chain is reported by generating an ECH interrupt.

It is also possible to abort the current chain of conversions by setting the MCR[ABORTCHAIN] bit. In that case, the behavior of the ADC depends on the MCR[MODE] bit (One-Shot/Scan) for Normal conversion. In One-Shot mode, MSR[NSTART] is automatically reset together with the MCR[ABORTCHAIN]. Otherwise, in Scan mode, a new chain is started, and the current chain is aborted. The EOC of the current aborted conversion is not generated, but an ECH interrupt is generated to signal the end of the chain. For Injected conversion, the behavior of ABORTCHAIN is same as for One-shot Normal conversion. In this case, MSR[JSTART] is automatically reset together with the MCR[ABORTCHAIN].

When an ABORTCHAIN is requested while an injected chain is running over a suspended normal chain, both injected and normal chains are aborted; both MSR[NSTART] and MSR[JSTART] are reset. ABORT and ABORTCHAIN requests are ignored during BCTU conversion.

34.6.2 Crosstriggering Unit interface

The Crosstriggering Unit (BCTU) interface enhances the injected conversion capability of the ADC. The BCTU contains multiple event inputs that can be used to select the channels to be converted from the appropriate event configuration register. The BCTU/ADC interface is shown in the figure below. The BCTU generates a trigger and a channel number to be converted. A single channel is converted for each request. After performing the conversion, the ADC returns the result.

The conversion result is also saved in the corresponding data register and it is compared with watchdog thresholds if requested. DMA and interrupt feature of ADC can also be used during CTU conversion.

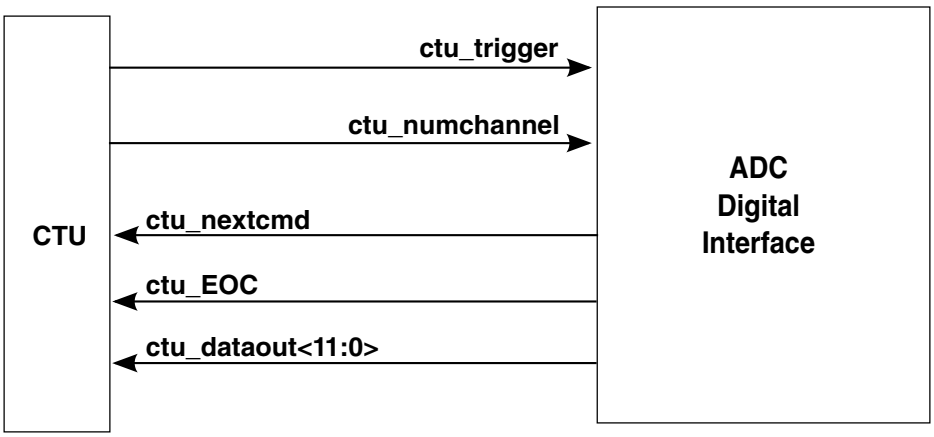


Figure 34-5. ADC Crosstriggering Unit

The BCTU interface has two modes of operation: Trigger and Control.

The BCTU interface is enabled by programming the MCR[CTUEN] bit. The BCTU interface operating mode is programmed by MCR[CTU_MODE] bit.

34.6.2.1 BCTU Trigger mode

In trigger mode, normal and injected conversions are enabled along with BCTU conversion, allowing all kinds of conversions to run.

When a trigger is received, the channel number is taken as the channel value and the BCTU conversion starts. The MSR[CTUSTART] bit is set automatically at this point and it is also automatically reset when the BCTU conversion is completed.

BCTU conversions should be requested after calibration is over. If a BCTU trigger is received during calibration, the calibration fails and BCTU conversion is started.

If a BCTU trigger is received during an ongoing injected conversion, then the injected chain is aborted immediately and only the BCTU conversion proceeds. The MSR[JSTART] bit is reset. Aborting of the injected conversion is indicated through the status bit of MSR[JABORT].

If a BCTU trigger is received during a ongoing normal conversion, the ongoing normal channel is suspended and the BCTU conversion is processed. Normal conversion resumes from the suspended channel after completion of BCTU triggered conversion.

If another BCTU trigger is received before ctu_nextcmd signal, that triggered conversion request is discarded.

If a Normal conversion is requested during BCTU conversion (MSR[CTUSTART] bit equal to ‘1’), the Normal conversion starts after the BCTU conversion is completed (MSR[CTUSTART] bit reset).

Injected conversion requests are discarded during a BCTU conversion, and the MCR[JSTART] bit is immediately reset.

34.6.2.2 BCTU Control mode

In BCTU Control mode, if enabled via MCR[CTUEN], the CPU is able to write in the ADC registers, but it cannot start a conversion. Conversion requests can be generated only by a BCTU trigger. If a normal or injected conversion is requested when the BCTU is enabled (MCR[CTUEN]), it will be automatically discarded.

Along with BCTU trigger the ctu_numchannel is taken as channel and BCTU conversion starts. MSR[CTUSTART]is set automatically at this point and it remains set unless the BCTU is disabled by resetting MCR[CTUEN].

BCTU conversions must be requested when calibration is over. If a BCTU trigger is received during calibration, the calibration stops immediately in order to satisfy the BCTU request. Calibration will fail in this case.

34.6.3 ADC clock prescaler and sample time settings

The clock (AD_clk) provided to the ADC's SAR controller, which controls the ADCA, must satisfy particular conditions of frequency and duty cycle. The maximum acceptable frequency is 80 MHz with a duty cycle equal to 50% (±5%).

The AD_clk frequency can be scaled by programming the MCR[ADCLKSEL] bit. If this bit is set, AD_clk frequency is the same as the bus clock. Otherwise, AD_clk frequency is half of the bus clock. MCR[ADCLKSEL] can only be written in power-down, when MCR[PWDN] = 1.

Sampling times are controlled by settings in the Conversion Timing Register (CTR*n*). A CTR register is provided for each channel type.

- CTR0 controls sampling time for *precision* channels:
  - 0 - 23 (ADC0—10 bit)
  - 0 - 15 (ADC1—12 bit)
- CTR1 controls sampling time for *internal* channels:

- 32 - 63 (ADC0—10 bit)
- 32 - 57 (ADC1—12 bit)

- CTR2 controls sampling time for *external* channels:
  - 64 to 95 (ADC0—10 bit)
  - 64 to 95 (ADC1—12 bit)

34.6.4 Presampling

Presampling allows the ADC internal capacitor to precharge or discharge to a defined level before it starts the actual sampling/conversion of the analog input coming from the pads. This is useful for resetting information (history effect/offset) regarding the last converted data. During presampling, the ADC samples the internally generated voltage while, during sampling, the ADC samples the analog input coming from the pads.

Presampling can be enabled/disabled on an individual channel by setting the corresponding bit in the applicable PSR*n* register.

- PSR0 controls precision channels
- PSR1 controls internal channels
- PSR2 controls external channels

After enabling the presampling for a channel, the normal sequence of operation is presampling + sampling + evaluation for that channel. Sampling of the channel can be bypassed by setting PSCR[PRECONV]. When sampling of a channel is bypassed, the presampled voltage will be converted.

See Conversion time for the conversion's timing equation.

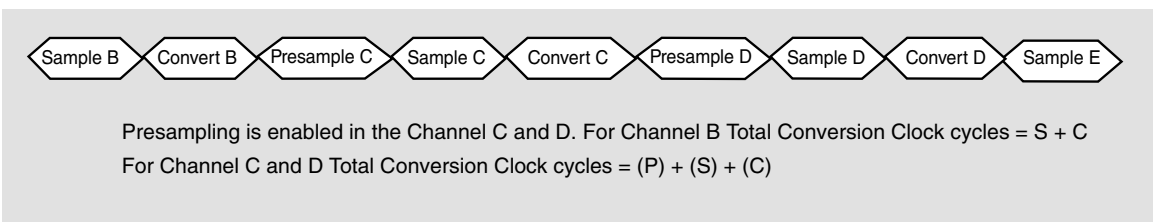


Figure 34-6. Presampling sequence

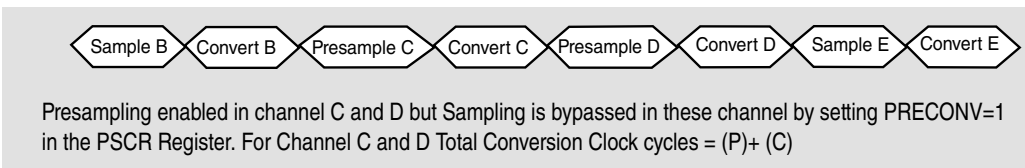


Figure 34-7. Presampling sequence with PRECONV = 1

34.6.4.1 Presampling channel enable

The presampling channels are selected by programming the presampling register (Presampling Control Register (ADC_PSCR)). These are used to enable analog switches that samples an internally generated voltage. It is possible to select presample voltage from four internally available voltages depending on the value of PREVAL*x* [*x* = 0,1,2] fields in the PSCR register as given in Table 34-7.

Table 34-7. Presample voltage selection

PREVAL*n*	Presampling voltage
00b	Presample voltage 1: VSS_HV_ADC0 (SAR ADC 0 ground), VSS_HV_ADC1 (SAR ADC 1 ground)
01b[1]	Presample voltage 2: VDD_HV_ADC0/8 (SAR ADC 0 supply divided by 8), VDD_HV_ADC1/8 (SAR ADC 1 supply divided by 8)
10b	Presample voltage 3: V_{REFL}
11b	Presample voltage 4: VDD_HV_ADC1_REF ADC1 (SAR ADC 1 reference high)

1. The voltage sampled inside ADCA is always (analog supply) / 8 for PSCR[PREVAL*n*] = 01b. However, when sampling is bypassed (PSCR[PRECONV] = 1), ADCD multiplies the conversion result by 8 to make the conversion result equal to the analog supply, or maximum value possible during saturation.

   So, for an open or short circuit test on any channel with presampling feature and PSCR[PRECONV] = 0b, user should expect conversion results as shown below:
   - Equal to Vin of the channel (if there is no open or short defect)
   - Close to (analog supply) / 8 if the channel is open (not connected to any voltage source, internally or externally)
   - Close to 0 V if the channel has a short to ground (analog ground, either internally or externally).
   - Close to the analog supply if the channel has a short to supply (analog supply, either internally or externally).

The PREVAL*n* fields are associated with specific channels, as shown in Table 34-8.

Table 34-8. PREVAL*n* channel association

Field	Channels
PREVAL0	• 0 - 23 (ADC0—10 bit) • 0 - 15 (ADC1—12 bit)
PREVAL1	• 32 - 63 (ADC0—10 bit) • 32 - 57 (ADC1—12 bit)
PREVAL2	• 64 to 95 (ADC0—10 bit) • 64 to 95 (ADC1—12 bit)

34.6.5 Programmable analog watchdog

NOTE

The number of analog watchdogs and channels may vary depending on the ADC instance on your chip. Please see the chip-specific section for how many are available on each instance.

The analog watchdogs are used for determining whether the result of a conversion lies within a given guard area (as shown in Figure 34-8) specified by an upper and a lower threshold value named THRH and THRL, respectively.

After the conversion of the selected channel, a comparison is performed between the converted value and the threshold values. If the converted value lies outside the guard area, then the corresponding threshold violation interrupt is generated.

The comparison result is stored as WTISR[WDGxH] and WTISR[WDGxL], as explained in the following table. Depending on the WTIMR[MSKWDGxL] and WTIMR[MSKWDGxH] mask bits, an interrupt is generated upon threshold violation.

Table 34-9. Values of WDGxH and WDGxL fields

WDGxH	WDGxL	Converted data
1	0	converted data > THRH
0	1	converted data < THRL
0	0	THRH ≤ converted data ≤ THRL

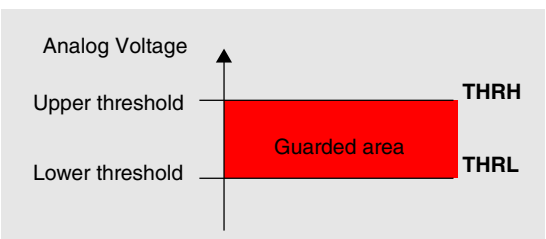


Figure 34-8. Guarded Area

34.6.5.1 Analog watchdog setting

For each watchdog there is one Threshold value (Low and High) Register (THRHLR).

The analog watchdog for each channel can be enabled independently by programming CWENR*n* register bits.

- CWENR0 controls *precision* channels:

    - 0 - 23 (ADC0—10 bit)
    - 0 - 15 (ADC1—12 bit)
- CWENR1 controls *internal* channels:
    - 32 - 63 (ADC0—10 bit)
    - 32 - 57 (ADC1—12 bit)
- CWENR2 controls *external* channels:
    - 64 to 95 (ADC0—10 bit)
    - 64 to 95 (ADC1—12 bit)

The availability of these registers depends on device configuration.

Threshold values for each channel can be selected independently from a maximum of 16 threshold registers (THRHLRx,(x=0..15)) by the WSEL_CHx field of CWSELRx (x=0..11) register.

Each CWSELRx register holds selection for 8 consecutive channels. CWSELR0 holds 8 WSEL_CH fields for channel 0 to 7. Likewise CWSELR1 is for channel 8 to 15 and so on. The availability of fields and registers depends on device configuration.

If the converted value for a particular channel lies outside the range specified by threshold values, then the corresponding bit is set in the Analog Watchdog Out of Range Register (AWORR).

For example, if channel number 12 is to be monitored with the threshold values in register THRHLR3, then CWSELR1[WSEL_CH4] needs to be programmed with 3. The enabling will be done by setting the bit corresponding to channel 12 in the CWENR0 register.

A set of threshold values (THRHLRx) can be linked to several ADC channels. The threshold values to be selected for a channel needs be programmed only once in the CWSELRx register.

34.6.6 DMA functionality

Conversion data of any channel can be transferred from register to system memory via Direct Memory Access (DMA). The DMA transfers can be enabled by setting ADC_DMAE[DMAEN]. Once enabled, the on chip DMA controller can get DMA requests after the conversion of every channel by setting the respective masking bit in the DMAR*n* registers. The DMA masking registers must be programmed before starting any conversion.

The DMA request to DMA controller can be cleared at different times in two modes:

MODE-1: Clearing of DMA request on Acknowledgement from DMA controller (ADC_DMAE [DCLR] = 0)

MODE-2: Clearing of DMA request on Read to data registers (ADC_DMAE [DCLR] = 1)

The figures below shows DMA operation in two modes (cycle counts are typical values).

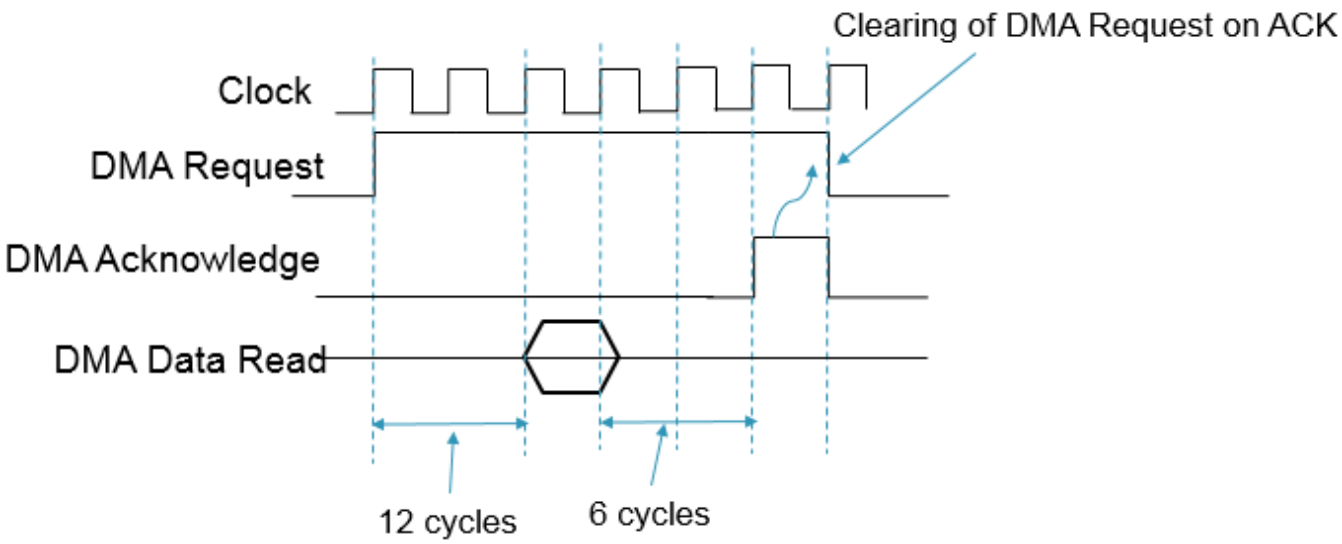


Figure 34-9. DMA operation MODE – 1 (DMAE [DCLR] = 0)

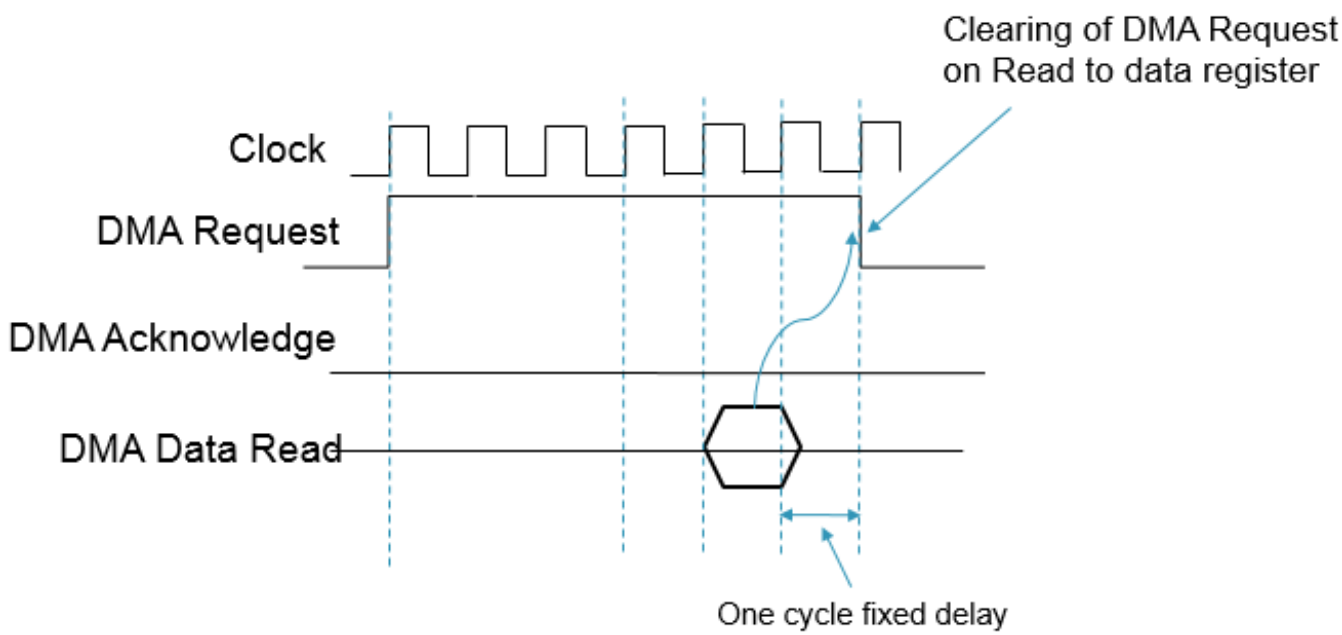


Figure 34-10. DMA operation MODE – 2 (DMAE [DCLR] = 1)

34.6.7 Interrupts

ADC generates the following maskable interrupt signals:

- End-of-conversion interrupt (EOC) request
- End-of-chain interrupt (ECH) request
- End-of-injected conversion interrupt (JEOC) request
- End-of-injected chain interrupt (JECH) request
- End-of-CTU conversion interrupt (EOCTU) request
- Watchdog threshold interrupt (WDGxL and WDGxH) requests

Interrupts are generated during the conversion process to signal events, like the end of a conversion, as explained in the register description for the Interrupt Status Register (ISR). The ISR and the Interrupt Mask Register (IMR) allow you to check and enable the interrupt request.

Interrupts can be individually enabled on a channel-by-channel basis by programming the Channel Interrupt Mask Register (CIMR).

Three Channel Pending Registers (CEOCFR*n*) are also provided in order to signal which of the channels' measurement has been completed.

Analog Watchdog interrupts are enabled by Watchdog Threshold Interrupt Mask Register (WTIMR) and the status can be checked at Watchdog Threshold Interrupt Status Register (WTISR). The watchdog interrupt source sets two fields, WDGxH and WDGxL, for each channel monitored in the WTISR.

In particular, the EOC, ECH, EOCTU, JEOC, and JECH interrupt lines are combined (OR-ed) on the same line. The EOC interrupt request is cleared by clearing either ISR[EOC] or all bits of CEOCFR. The JEOC interrupt request is cleared by clearing either ISR[JEOC] or all bits of CEOCFR. The EOCTU interrupt request is cleared by clearing either ISR[EOCTU] or all bits of CEOCFR.

All WDGxH and WDGxL requests are combined on the second line.

Another interrupt on the third line is a combination of WDSERR, WDTERR, ERR_S0, ERR_S1, and ERR_S2.

The ISR register contains the interrupt pending request status. If you want to clear a particular interrupt event status, then writing a 1 to the corresponding status bit will clear the pending interrupt flag. (At this write operation, all the other bits of the ISR register must be programmed to 0.)

34.6.8 External Decode signals Delay

The ADC provides three external decode signals to be used to select one channel out of eight channels in external Analog multiplexers. There can be a maximum of four such multiplexers that are used for the 32 external channels to be converted. In order to take into account the control of switching time of the external analog mux, a Decode Signals Delay (DSD) register is provided. This register allows you to program the delay between the decoding signal selection and the actual start of conversion.

If presampling is enabled then the DSD delay starts after the presample phase (presampling time is same as sample time).

34.6.9 Power-Down mode

The analog part of the ADC can be put in Power-Down mode by setting the MCR[PWDN] bit, which shuts off the ADCA and stops the clock in the SAR controller. After power-on or device reset, the ADCA is kept in Power-Down mode by default, so this state must be exited before starting any operation by resetting the MCR[PWDN] bit. In Power-Down mode, no conversion and calibration can be started. Trigger from BCTU during power down is discarded. In the ADC power-down mode, the BCTU must not start any conversion.

The MCR[PWDN] bit may be programmed high at any time. However, if a conversion is ongoing, then the ADCA cannot be sent immediately into Power-down mode. In fact, the ADC enters Power-down mode only after completing the ongoing conversion. Otherwise, the ongoing operation should be aborted manually by resetting the NSTART bit and using the ABORTCHAIN function.

The ADC status in MSR[ADCSTATUS] field shows Power-down status only when ADC enters in Power-down mode.

If BCTU is enabled and MSR[CTUSTART] is 1, then the MCR[PWDN] bit cannot be set. When BCTU Control mode is enabled, the application needs to reset the CTUEN bit before entering Power-Down mode.

After the power-down phase is complete, the process ongoing before the power-down phase must be restarted manually .

NOTE

Resetting the MCR[PWDN] bit and setting the MCR[NSTART] or MCR[JSTART] bit during the same cycle is forbidden.

34.6.10 Auto-Clock-Off mode

To reduce power consumption in Idle mode (without going into Power-Down mode), an auto-clock-off feature can be enabled by setting MCR[ACKO]. When enabled, the analog clock is automatically switched off when no operation is ongoing (that is, when no conversion has been programmed).

NOTE

This feature must remain off during calibration.

34.6.11 Calibration

The ADC is calibrated at runtime. The test result is used for computation and is stored during the calibration process.

Calibration test mode is referred to as *High-Accuracy* or *Calibration* mode, and needs to be run after power-on reset and whenever required in runtime operation. This mode calibrates the ADC and updates calibration-related registers.

The following section describes the steps for executing calibration.

34.6.11.1 Calibration (High Accuracy mode)

The functioning of the analog block of ADC depends on the accuracy of values of its various parts and parameters, such as the value of capacitors, the gain of the amplifier, and so on. Due to variations in manufacturing and various runtime environmental effects, the actual values may differ from the ideal values for which the ADC is targeted.

Because of these deviations, the ADC's converted values contain errors. To eliminate these errors, the ADC must be calibrated prior to any conversion.

In the calibration process, a fixed known reference voltage (VrefH) is sampled many times and converted under various controlled conditions to check for deviations between these converted values and predefined values.

The deviations, known as offsets or modified values, are used to eliminate errors during the data processing of normal conversions.

Note

The ADC must be recalibrated if the operating conditions (particularly VrefH) change significantly.

Recommended settings:

- Clock frequency = 40 MHz
- Averging – Maximum (512)

The calibration process will take ~12ms of time with the above recommended settings.

NOTE

If the ADC clock frequency is > 40MHz during calibration, the accuracy of the calibration process will be compromised, hence the performance of the ADC. Thus, it is strongly recommended to run the calibration with 40 MHz clock.

As calibration is a time consuming process, it can be performed after a long interval – probably several hours of operation – but it is not necessary to recalibrate frequently. The goal is to maximize the possible accuracy delivered from conversion and to catch any structural fault to flag the error.

The calibration routine sets the fail flag (CALBISTREG[TEST_FAIL]) to indicate:

- A fault in ADC
- A premature abort of the calibration

Calibration is aborted if a normal conversion is initiated during calibration; this is called *premature termination*. The terminating conversion and the immediate next one to it will have invalid result in this case.

CALBISTREG[TEST_FAIL] = 1 indicates that the ADC health is not good and that it will not meet specifications.

To execute this routine, follow these steps:

1. Configure the ADC to 40 MHz.
   - Put the ADC in power down mode: set MCR[PWDN]
   - Configure clock divider selection: reset MCR[ADCLKSEL]
   - Power-up the ADC: reset MCR[PWDN]
2. Configure the Calibration, BIST Control, and Status Register (CALBISTREG). The default values are set for maximum accuracy (recommended).
3. Start calibration: set CALBISTREG[TEST_EN].
4. Wait for calibration to complete. Check MSR[ADCSTATUS] or till CALBISTREG[C_T_BUSY] becomes 0.
5. Check the value of MSR[CALIBRTD] to determine whether calibration was successful.
6. If MSR[CALIBRTD] = 1, calibration was successful; otherwise, it failed, and CALBISTREG[TEST_FAIL] = 1.
7. If required, reconfigure the ADC clock frequency for normal operation.

34.6.12 Conversion time

Total conversion time depends on ADC-SAR controller's operating clock. This clock can be prescaled to bus clock ÷ 2. The ADC-SAR operating clock (AD_clk) is always synchronous with bus clock. The frequency of AD_clk is dependent on settings of ADC_MCR[ADCLKSEL].

If ADC_MCR[ADCLKSEL] = 0, then AD_clk = (bus clock ÷ 2)

If ADC_MCR[ADCLKSEL] = 1, then AD_clk = bus clock

The various components of conversion time and affecting configurations are shown and described in Figure 34-11.

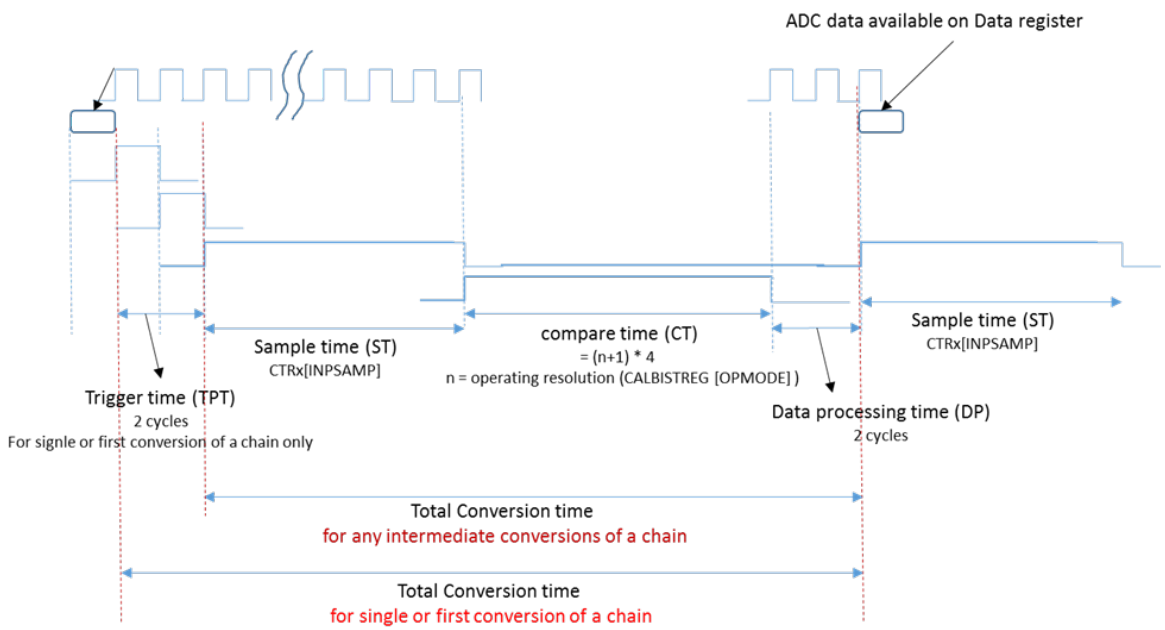


Figure 34-11. Conversion time

- Trigger processing time (TPT):
  To prepare the channel and start the operation, two clock cycles of bus clock is required by the ADC for the first conversion and for subsequent conversions in a chain, or for a continuous conversion, this time is not required as it is hidden in the pipeline operation. Triggers from Synchronous CTU interface requires two cycle of bus clock for first conversion and for back-to-back CTU conversions it takes only one cycle for trigger processing as the other cycle is hidden in pipeline operation. Triggers from Asynchronous CTU interface requires three cycles of bus clock for synchronization and processing.
- Sample phase time (ST):

Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing Registers ADC_CTR*x* (*x* = 0...2) for different types of channels. Value in each register affects the sample time in similar way. The value in the register represents units of cycle of the AD_clk. The minimum value of sample time is 8. If the value programmed is less than 8, then it has no effect on sample time duration and in this case sample time will be 8 AD_clk cycles.

- Compare phase time (CT):

  The compare phase of the ADC dependends on the resolution setting (which is 13 for this chip), and takes ((13 + 1) × 4) cycles of AD_clk.

- Data processing time (DP):

  The ADC takes 2 cycles of AD_clk to post process and load the data int registers.

- Total Conversion Time:

  The total time required for single or first conversion is [TPT+ST+CT+DP] cycles of AD_clk and for subsequent conversion is a chain or continuous conversion is [ST+CT+DP] cycles of AD_clk.

  Examples:

  - CTR*x*[INPSAMP] = 20d; software/hardware triggered (NSTART):
    - Single/First Conversion: 2 + 20 + ((13 + 1) × 4) + 2 = 2 + 20 + 56 + 2 = 80 cycles of AD_clk
    - Subsequent/continuous conversion: 20 + ((13 + 1) × 4) + 2 = 20 + 56 + 2 = 78 cycles of AD_clk
  - CTR*x*[INPSAMP] = 20d; CTU triggered:
    - Single/First Conversion: 2 + 20 + ((13 + 1) × 4) + 2 = 2 + 20 + 56 + 2 = 80 cycles of AD_clk
    - Subsequent/continuous conversion: 1 + 20 + ((13 + 1) × 4) + 2 = 1 + 20 + 56 + 2 = 79 cycles of AD_clk

- Pre-Sampling:

  When pre-sampling is enabled, the ADC does pre-sampling prior to starting the actual channel sampling as described in pre-sampling section. This phase takes CTR*x*[INPSAMP] cycles of AD_clk. Additionally, to switch from pre-sample phase to actual channel sampling phase, it takes another two cycles of AD_clk. So, total CTR*x*[INPSAMP] + 2 cycles additionally required for a conversion to be completed. Thus, in case of pre-sample, all the equations and examples above are valid with the addition of (CTR*x*[INPSAMP] + 2) cycles.

34.6.12.1 Conversion time for BCTU interface

The BCTU can work in a different clock domain, and for this there is an asynchronous domain crossing that occurs before the CTU trigger is processed. Data is also processed after domain crossing. These times need to be considered in the conversion time calculation. The synchronization may take a maximum of two additional ADC clock cycles and three BCTU clock cycles over normal conversion time.

The total conversion time taking BCTU timing into consideration is as follows:

Total conversion time = Synchronization time + TPT + ST + CT + DP + Reverse synchronization time in BCTU

The figure below shows the different components of total conversion time.

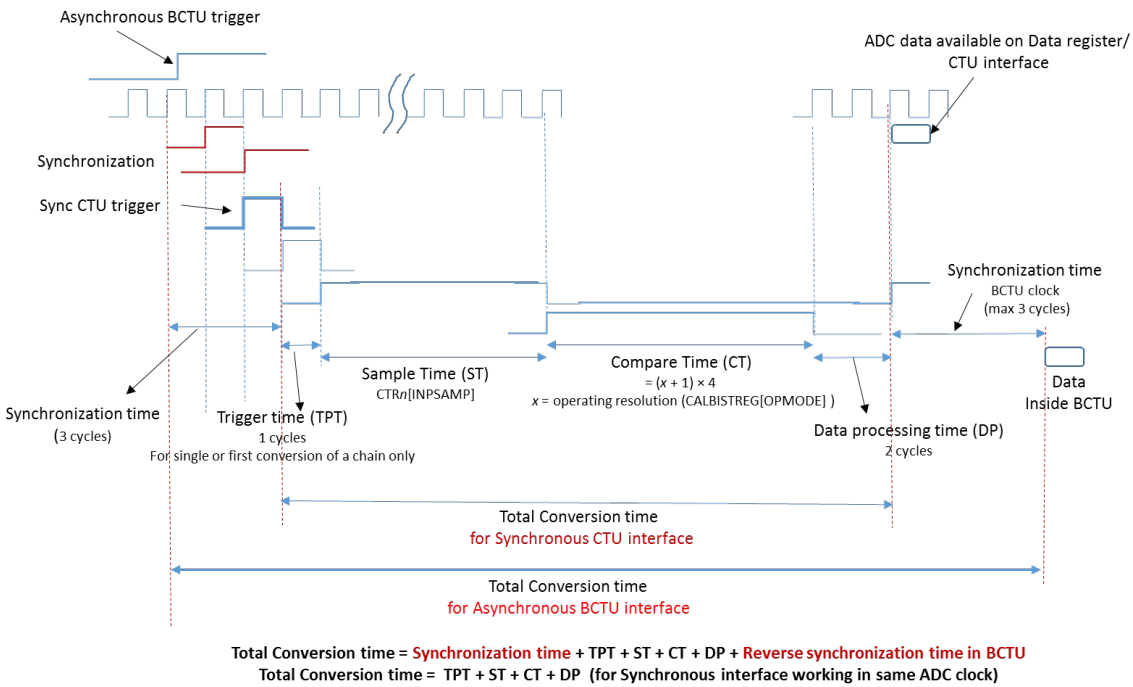


Figure 34-12. Normal Conversion time from BCTU (asynchronous, not working in ADC clock)

34.6.13 Conversion data processing

The raw converted ADC data contains many types of errors such as offset, gain, DC bias, and so on. Thus, to generate error-free results, raw converted data is processed before it is written to a result register. The process of error correction goes bit-by-bit during conversion with the values generated during the offset calculation and calibration process. This helps in reducing final data processing time as only one addition/addition (2's complement) is performed simultaneously with conversion.

34.7 Clock frequency

The ADC internal blocks are designed to work with an 80 MHz clock. Thus, the specified speed can be met with this clock frequency. If the clock is faster than this, ADCA cannot work properly and the ADC may malfunction. If the clock is slower, then the ADC speed will be lower, but it will work properly. It should be noted that there is an absolute maximum value of the sampling time that the analog portion of the ADC can support. For slower clocks, the sampling clock count will need to be programmed accordingly.

Chapter 35
Analog Comparator (CMP)

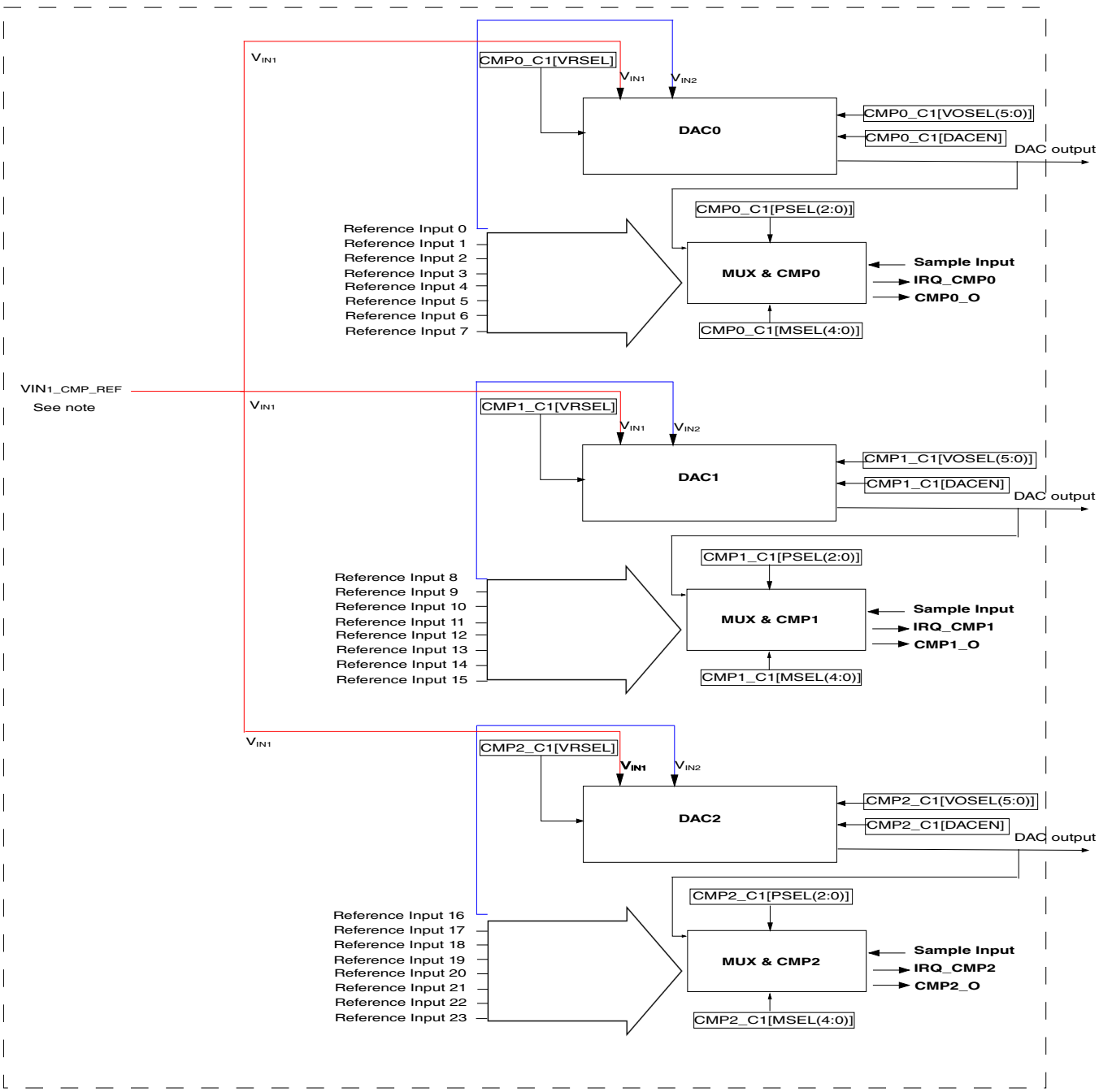
35.1 Chip-specific CMP information

35.1.1 CMP configuration

This device supports three instances of this module: CMP0, CMP1, and CMP2. This gives a total of 24 external inputs. In addition, each comparator has its own unique DAC reference. Thus, each comparator has a total of nine inputs.

The DAC in each comparator is a 6-bit comparator, yielding 64 reference points.

The MDIS/PCTL control bits will control the ACMP bus clock in various mode. The comparison will be active even when bus clock is disabled.



Note: $V_{IN1_CMP_REF}$ is not available in the LQFP176 SO V_{IN1} is connected to $V_{DD_HV_A}$ in LQFP176.

Figure 35-1. Comparator connectivity with 24 channel inputs

35.1.2 CMP input output connections

The following table shows the fixed connections to the CMP.

Table 35-1. CMP input output connections

Source	CMP0	CMP1	CMP2
IN0	CMP0_0	CMP1_8	CMP2_16
IN1	CMP0_1	CMP1_9	CMP2_17
IN2	CMP0_2	CMP1_10	CMP2_18
IN3	CMP0_3	CMP1_11	CMP2_19
IN4	CMP0_4	CMP1_12	CMP2_20
IN5	CMP0_5	CMP1_13	CMP2_21
IN6	CMP0_6	CMP1_14	CMP2_22
IN7	CMP0_7	CMP1_15	CMP2_23
OUT	CMP0_O	CMP1_O	CMP2_O

35.1.3 CMP "V_{IN1}" and "V_{IN2}" references

The 6-bit DAC sub-block supports selection of "V_{IN1}" and "V_{IN2}" references.

For each comparator, one of the references ("V_{IN1},V_{IN2}") for DAC has a dedicated pin $V_{IN1_CMP_REF}$ and the other source is shared with input channel 0. Therefore, in order to use the other reference, channel 0 of the corresponding comparator has to be sacrificed. The reference switching is based on the VRSEL signal. So the comparators have to work on the external reference $V_{IN1_CMP_REF}$ to utilize all the available input channels otherwise one less than total number of channels and use internal reference – that is channel 0.

The following tables list the package-wise V_{IN1} and V_{IN2} references.

Table 35-2. CMPx package-wise Vin2 references

Package	CMP0_Vin2	CMP1_Vin2	CMP2_Vin2
All	CMP0_0	CMP1_8	CMP2_16

Table 35-3. CMP package-wise Vin1 references

Package	CMPx_Vin1
BGA324	$V_{IN1_CMP_REF}$
BGA256	$V_{IN1_CMP_REF}$
LQFP176	$V_{DD_HV_A}$

35.1.4 Interaction with RTC API to cause wakeup

The CMP module is used in conjunction with the RTC/API module to cause wakeup in STOP, LPU_STOP, STANDBY and LPU_STANDBY modes. The API is configured to generate a trigger signal to inform the CMP module to enable the DAC. After a short interval, the RTC/API sends the CMP module a START signal that begins the round robin operation. Following the START, the CMP module performs a round robin on all eight inputs while comparing each input with the fixed DAC voltage. On completion of the round robin, the result is compared to the previous stored result. If any difference is detected, the CMP module sends an interrupt to the system. The device comes out of the low power state via the wakeup module. The following figure shows a flowchart describing the procedure.

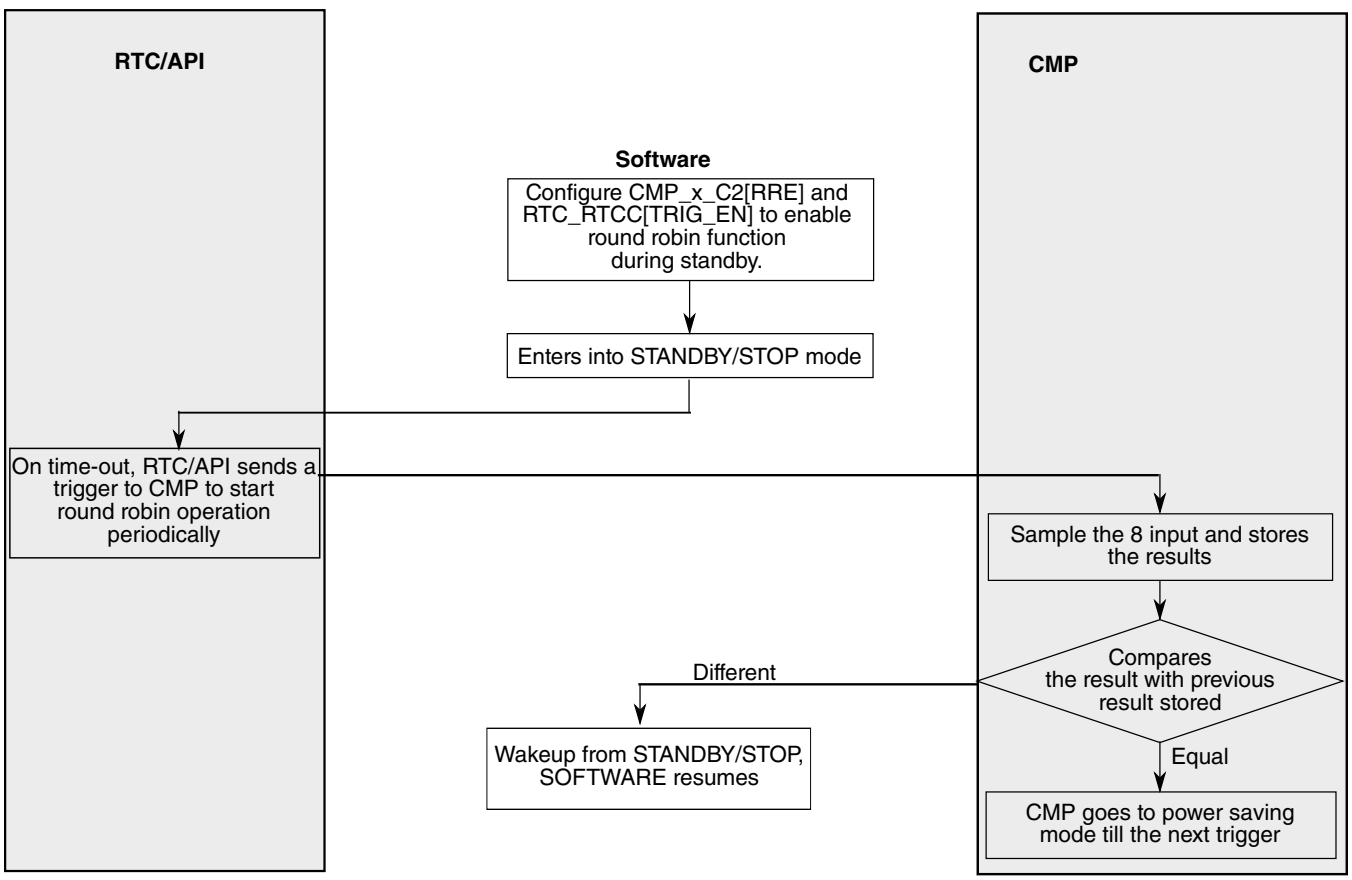


Figure 35-2. CMP-RTC interaction

In the round robin mode, only the continuous mode is supported. None of the window/ filter/sample functions should be used.

Summary of register configurations before entering STANDBY mode:

1. Configure CMP_x_C2[RRE] and RTC_RTCC[TRIG_EN] to enable round robin function during standby.
2. Enable the round robin interrupt by configuring RRIE in CMP_*x*_C2.
3. Enable the DAC to be used as the reference voltage on either plus or minus of the mux inside CMP.
4. Configure the NSAM field of the CMP_*x*_C2 register(s) to select the number of clock cycles for which a channel is selected for sampling.
5. Set ACO*n* for the predefined values of the state of the 8 bits corresponding to the eight inputs.
6. Configure RTC_APIVAL to set the period of the round robin operation.
7. Configure RTC_ANLCMP_CNT as the initialization time of the CMP module.

NOTE

This RTC API - CMP interaction for wakeup feature is available in STOP, LPU_STOP, STANDBY and LPU_STANDBY modes.

35.1.5 CMP Window control

The window in the window mode of the CMP is controlled via the GPR register bits in the device. If the filter is enabled and if the RTC clock is to be used as the clock to the filter block instead of the bus clock, the window mode should be disabled.

Whenever there is switching on bit GPR_CTL[CMP_SAMP_SEL], both the filter and window blocks needs to be disabled.

Refer to GPR_CTL[CMP_ACT_SAM] and GPR_CTL[CMP_SAMP_SEL] for details.

35.2 Introduction

The comparator (CMP) module provides a circuit for comparing two analog input voltages. The comparator circuit is designed to operate across the full range of the supply voltage, known as rail-to-rail operation.

The Analog MUX (ANMUX) provides a circuit for selecting an analog input signal from eight channels. One signal is provided by the 6-bit digital-to-analog converter (DAC). The mux circuit is designed to operate across the full range of the supply voltage.

The DAC is a 64-tap resistor ladder network that provides a selectable voltage reference for applications requiring a voltage reference. The 64-tap resistor ladder network divides the supply reference V_{in} into 64 voltage levels. A 6-bit digital signal input selects the

output voltage level, which varies from V_{in} to $V_{in}/64$. V_{in} can be selected from two voltage sources, V_{in1} and V_{in2}. The DAC from a comparator is available as an on-chip internal signal only and is not available externally to a pin.

35.3 Features

The following subsections list the features of the CMP, the DAC, and the ANMUX.

35.3.1 CMP features

The CMP has the following features:

- Operational over the entire supply range
- Inputs may range from rail to rail
- Programmable hysteresis control
- Selectable interrupt on rising-edge, falling-edge, or both rising or falling edges of the comparator output
- Selectable inversion on comparator output
- Capability to produce a wide range of outputs such as:
  - Sampled
  - Windowed, which is ideal for certain PWM zero-crossing-detection applications
  - Digitally filtered:
    - Filter can be bypassed
    - Can be clocked via external SAMPLE signal or scaled bus clock
- External hysteresis can be used at the same time that the output filter is used for internal functions
- Two software selectable performance levels:
  - Shorter propagation delay at the expense of higher power
  - Low power, with longer propagation delay
- DMA transfer support
  - A comparison event can be selected to trigger a DMA transfer

- Functional in all MCU power modes
- The window and filter functions are not available in STOP modes
- The comparator can be triggered by other peripherals to work for only a small fraction of the time

35.3.2 6-bit DAC key features

The 6-bit DAC has the following features:

- 6-bit resolution
- Selectable supply reference source
- Power Down mode to conserve power when not in use
- Option to route the output to internal comparator input

35.3.3 ANMUX key features

The ANMUX has the following features:

- Two 8-to-1 channel MUXes
- Operational over the entire supply range

35.4 CMP, DAC, and ANMUX diagram

The following figure shows the block diagram for the High-Speed Comparator, DAC, and ANMUX modules.

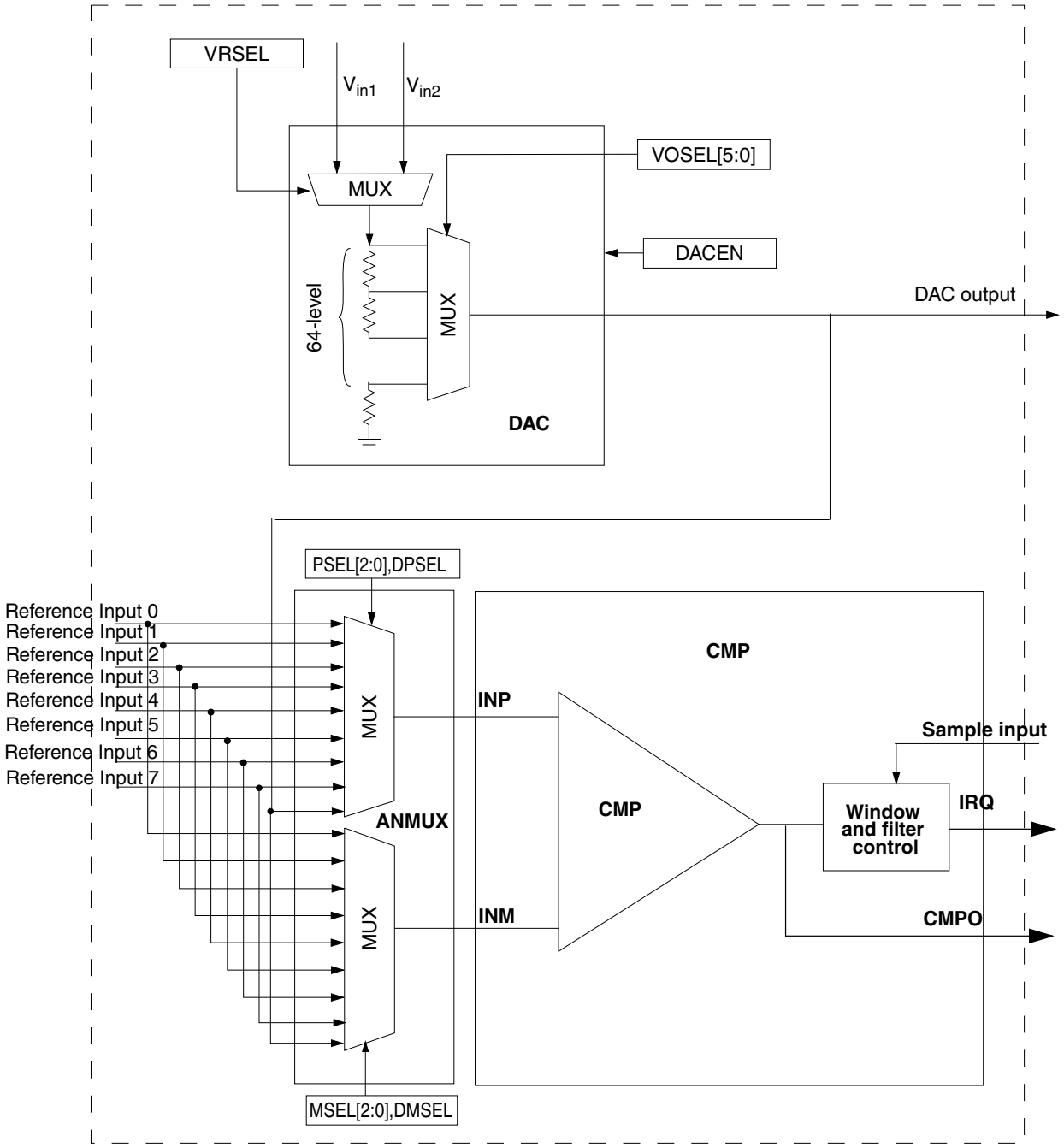


Figure 35-3. CMP high level diagram

35.5 CMP block diagram

The following figure shows the block diagram for the CMP module.

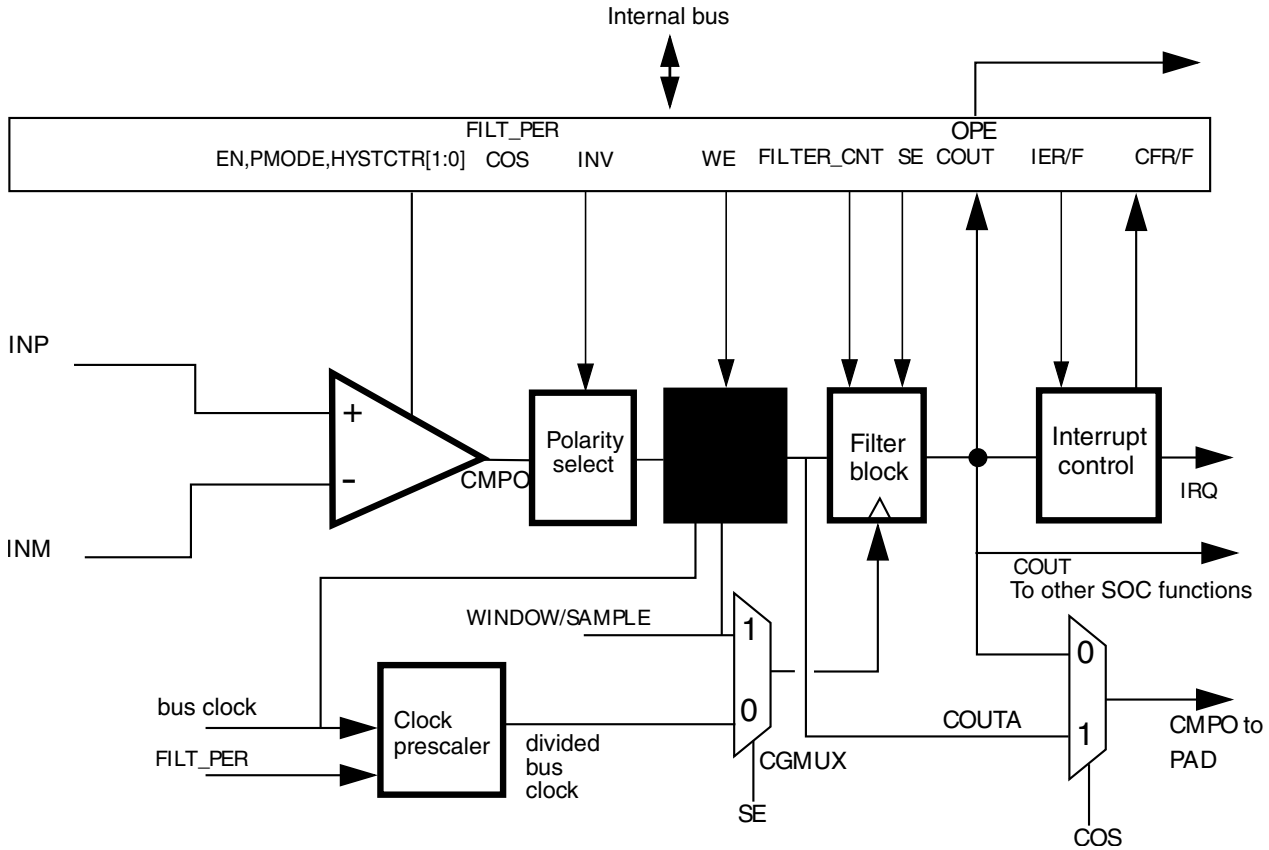


Figure 35-4. Comparator module block diagram

In the CMP block diagram:

- The Window Control block is bypassed when C0[WE] = 0.
- If C0[WE] = 1, the comparator output is sampled on every bus clock when WINDOW=1 to generate COUTA. Sampling does NOT occur when WINDOW = 0.
- The Filter block is bypassed when not in use.

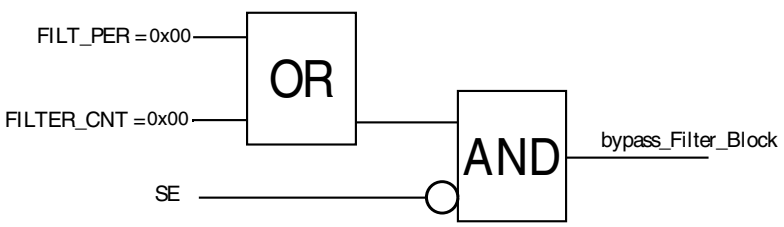


Figure 35-5. Filter block bypass logic

- The Filter block acts as a simple sampler if the filter is bypassed and C0[FILTER_CNT] is set to 0x01.
- The Filter block filters based on multiple samples when the filter is bypassed and C0[FILTER_CNT] is set greater than 0x01.

- If C0[SE] = 1, the external SAMPLE input is used as the sampling clock.
- IF C0[SE] = 0, the divided bus clock is used as the sampling clock.

- If enabled, the Filter block will incur up to one bus clock additional latency penalty on COUT due to the fact that COUT, which crosses clock domain boundaries, must be resynchronized to the bus clock.
- C0[WE] and C0[SE] are mutually exclusive.
- If enabled, the filter clock and the sample period must be at least 4 tims slower than the system clock to the comparator.

35.6 CMP pin descriptions

This section provides the comparator pin descriptions. The MSB input IN[7] is muxed with the DAC output on both the plus and minus sides by CMP_C1[DPSEL] and CMP_C1[DMSEL], respectively. When CMP_C1[DPSEL] is set, the DAC output is selected as the reference input for the plus MUX channel 7, regardless of the CMP_C1[PSEL] setting. Similarly, when CMP_C1[DMSEL] is set, the DAC output is selected as the reference input for the minus MUX channel 7, regardless of the CMP_C1[MSEL] setting.

Table 35-4. CMP signal descriptions

Signal	Description	I/O
IN[7:0]	Analog voltage inputs	I

35.6.1 External pins

The CMP has two analog inputs: INP and INM. Each of these pins can accept an input voltage that varies across the full operating range of the MCU. If the module is not enabled, each pin can be used as a digital input or output. Consult the specific MCU documentation to determine what functions are shared with these analog inputs.

The user can select either filtered or unfiltered comparator outputs for use on an external I/O pad.

35.7 CMP functional modes

There are three main sub-blocks to the CMP module:

- The comparator itself
- The window function
- The filter function

The filter, C0[FILTER_CNT], can be clocked from an internal or external clock source. The filter is programmable with respect to the number of samples that must agree before a change in the output is registered. In the simplest case, only one sample must agree. In this case, the filter acts as a simple sampler.

The external sample input is enabled using C0[SE]. When set, the output of the comparator is sampled only on rising edges of the sample input.

The "windowing mode" is enabled by setting C0[WE]. When set, the comparator output is sampled only when WINDOW=1. This feature can be used to ignore the comparator output during time periods in which the input voltages are not valid. This is especially useful when implementing zero-crossing-detection for certain PWM applications.

The comparator filter and sampling features can be combined as shown in the following table. Individual modes are discussed below.

Table 35-5. Comparator sample/filter controls

Mode #	C0[EN]	C0[WE]	C0[SE]	C0[FILTER_CNT]	C0[FPR]	Operation
1	0	X	X	X	X	**Disabled** See the Disabled mode (# 1).
2A	1	0	0	0x00	X	**Continuous Mode** See the Continuous mode (#s 2A & 2B).
2B	1	0	0	X	0x00	
3A	1	0	1	0x01	X	**Sampled, Non-Filtered mode** See the Sampled, Non-Filtered mode (#s 3A & 3B).
3B	1	0	0	0x01	> 0x00	
4A	1	0	1	> 0x01	X	**Sampled, Filtered mode** See the Sampled, Filtered mode (#s 4A & 4B).
4B	1	0	0	> 0x01	> 0x00	
5A	1	1	0	0x00	X	**Windowed mode**
5B	1	1	0	X	0x00	

Table continues on the next page...

Table 35-5. Comparator sample/filter controls (continued)

Mode #	C0[EN]	C0[WE]	C0[SE]	C0[FILTER_CNT]	C0[FPR]	Operation
						Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA. See the Windowed mode (#s 5A & 5B).
6	1	1	0	0x01	0x01–0xFF	**Windowed/Resampled mode** Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA, which is then resampled on an interval determined by C0[FPR] to generate COUT. See the Windowed/Resampled mode (# 6).
7	1	1	0	> 0x01	0x01–0xFF	**Windowed/Filtered mode** Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA, which is then resampled and filtered to generate COUT. See the Windowed/Filtered mode (#7).
All other combinations of C0[EN], C0[WE], C0[SE], C0[FILTER_CNT], and C0[FPR] are illegal.						

For cases where a comparator is used to drive a fault input, for example, for a motor-control module such as FTM, it must be configured to operate in Continuous mode so that an external fault can immediately pass through the comparator to the target fault circuitry.

Note

Filtering and sampling settings must be changed only after setting C0[SE]=0 and C0[FILTER_CNT]=0x00. This resets the filter to a known state.

35.7.1 Disabled mode (# 1)

In Disabled mode, the analog comparator is non-functional and consumes no power. CMPO is 0 in this mode.

35.7.2 Continuous mode (#s 2A & 2B)

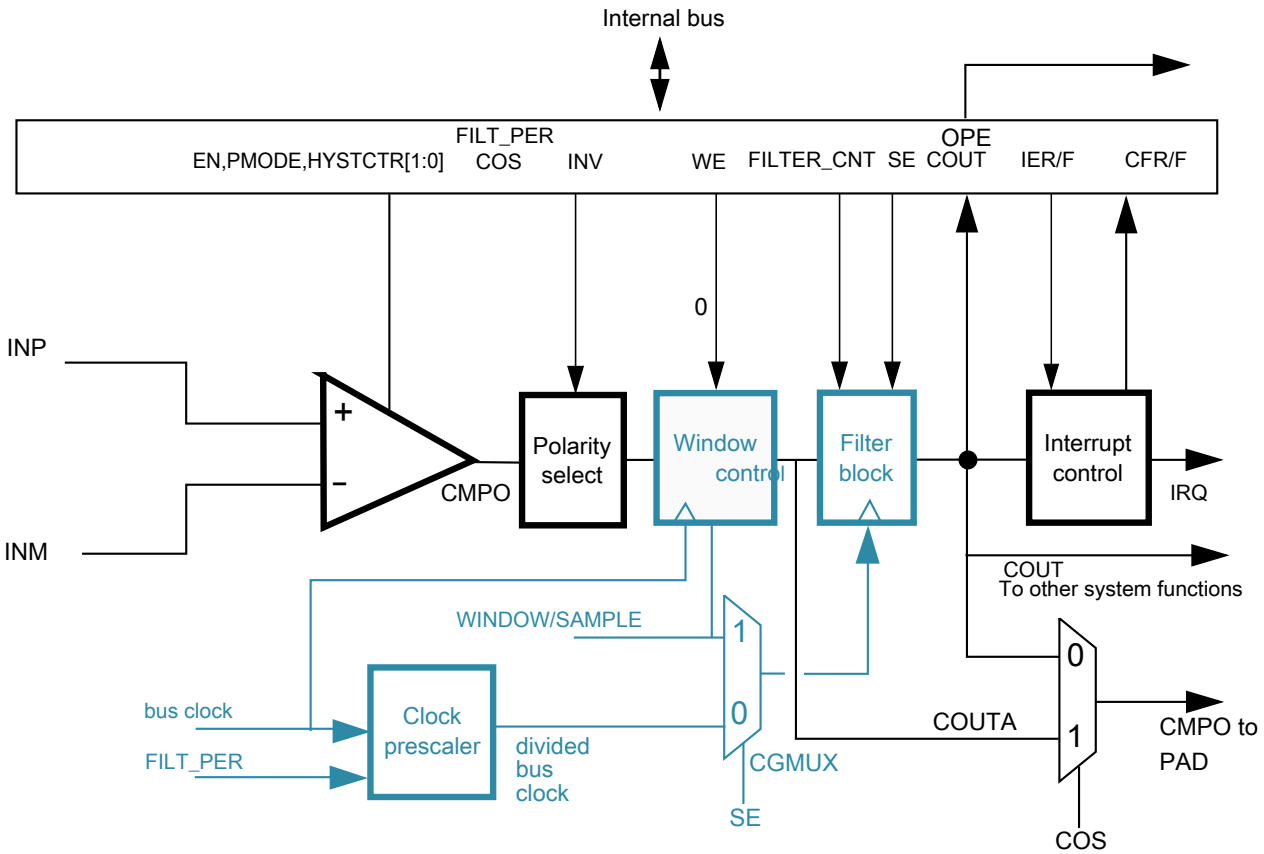


Figure 35-6. Comparator operation in Continuous mode

NOTE

See the chip configuration section for the source of sample/ window input.

The analog comparator block is powered and active. CMPO may be optionally inverted, but is not subject to external sampling or filtering. Both window control and filter blocks are completely bypassed. C0[COUT] is updated continuously. The path from comparator input pins to output pin is operating in combinational unclocked mode. COUT and COUTA are identical.

For control configurations that result in disabling the filter block, see Figure 35-5.

35.7.3 Sampled, Non-Filtered mode (#s 3A & 3B)

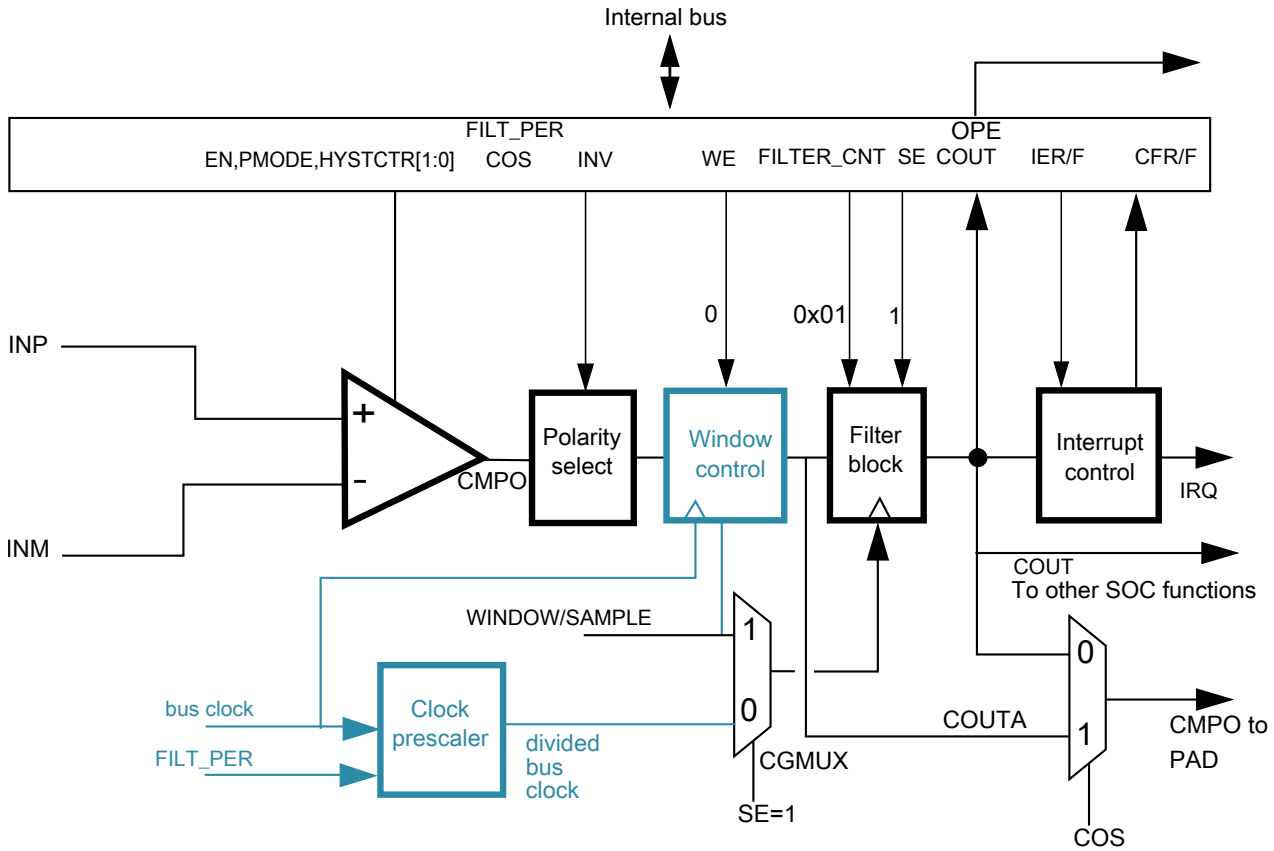


Figure 35-7. Sampled, Non-Filtered (# 3A): sampling point externally driven

In Sampled, Non-Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational unclocked. Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the filter block clock input.

The only difference in operation between Sampled, Non-Filtered (# 3A) and Sampled, Non-Filtered (# 3B) is in how the clock to the filter block is derived. In #3A, the clock to filter block is externally derived while in #3B, the clock to filter block is internally derived.

The comparator filter has no other function than sample/hold of the comparator output in this mode (# 3B).

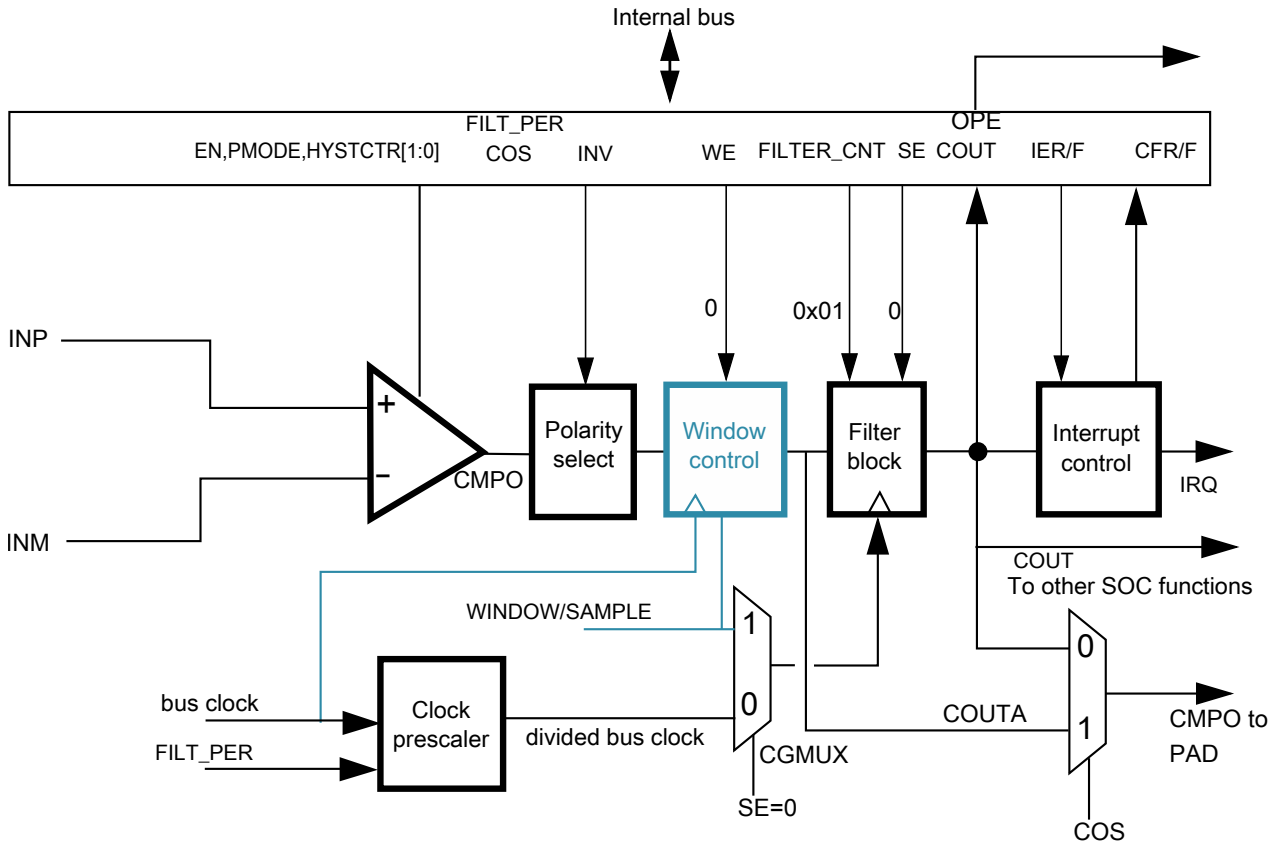


Figure 35-8. Sampled, Non-Filtered (# 3B): sampling interval internally derived

35.7.4 Sampled, Filtered mode (#s 4A & 4B)

In Sampled, Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational unclocked. Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the filter block clock input.

The only difference in operation between Sampled, Non-Filtered (# 3A) and Sampled, Filtered (# 4A) is that, now, C0[FILTER_CNT]>1, which activates filter operation.

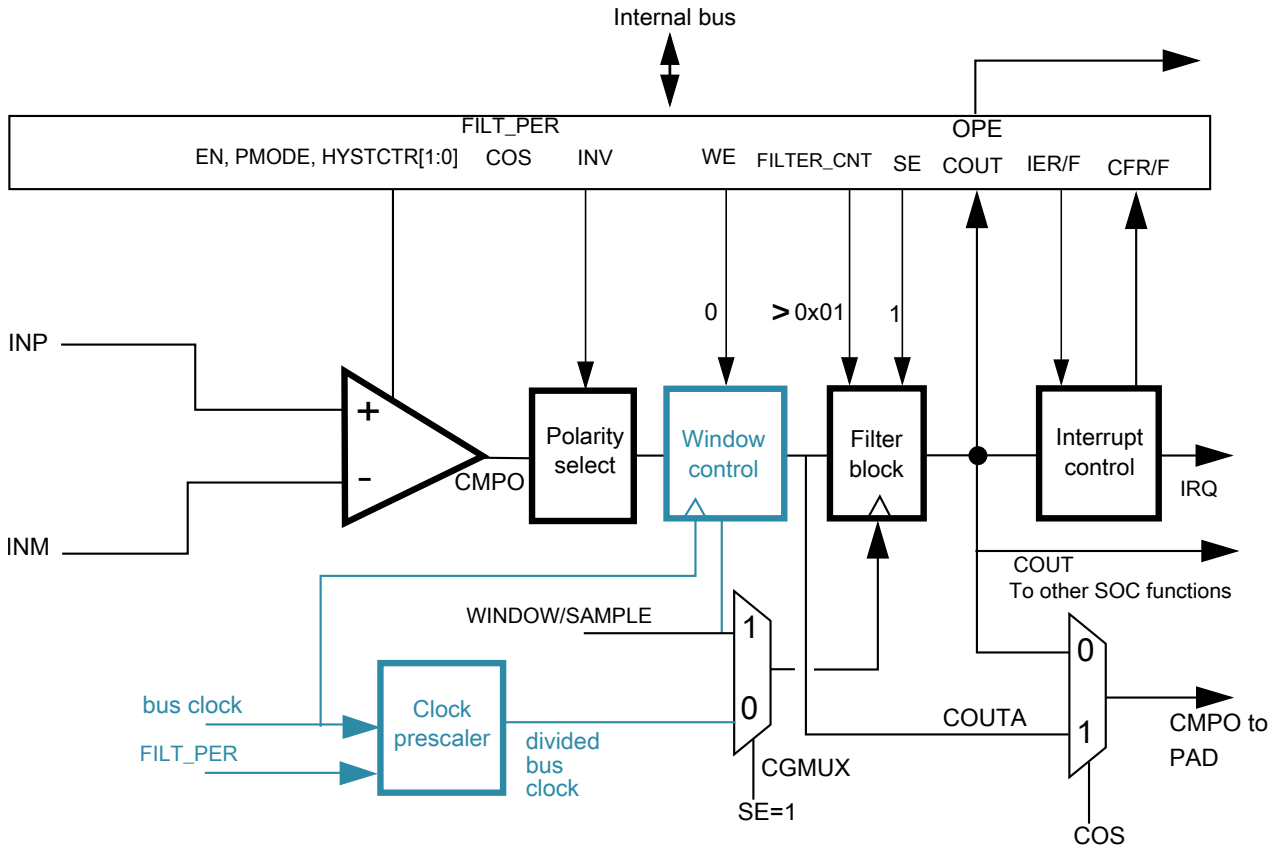


Figure 35-9. Sampled, Filtered (# 4A): sampling point externally driven

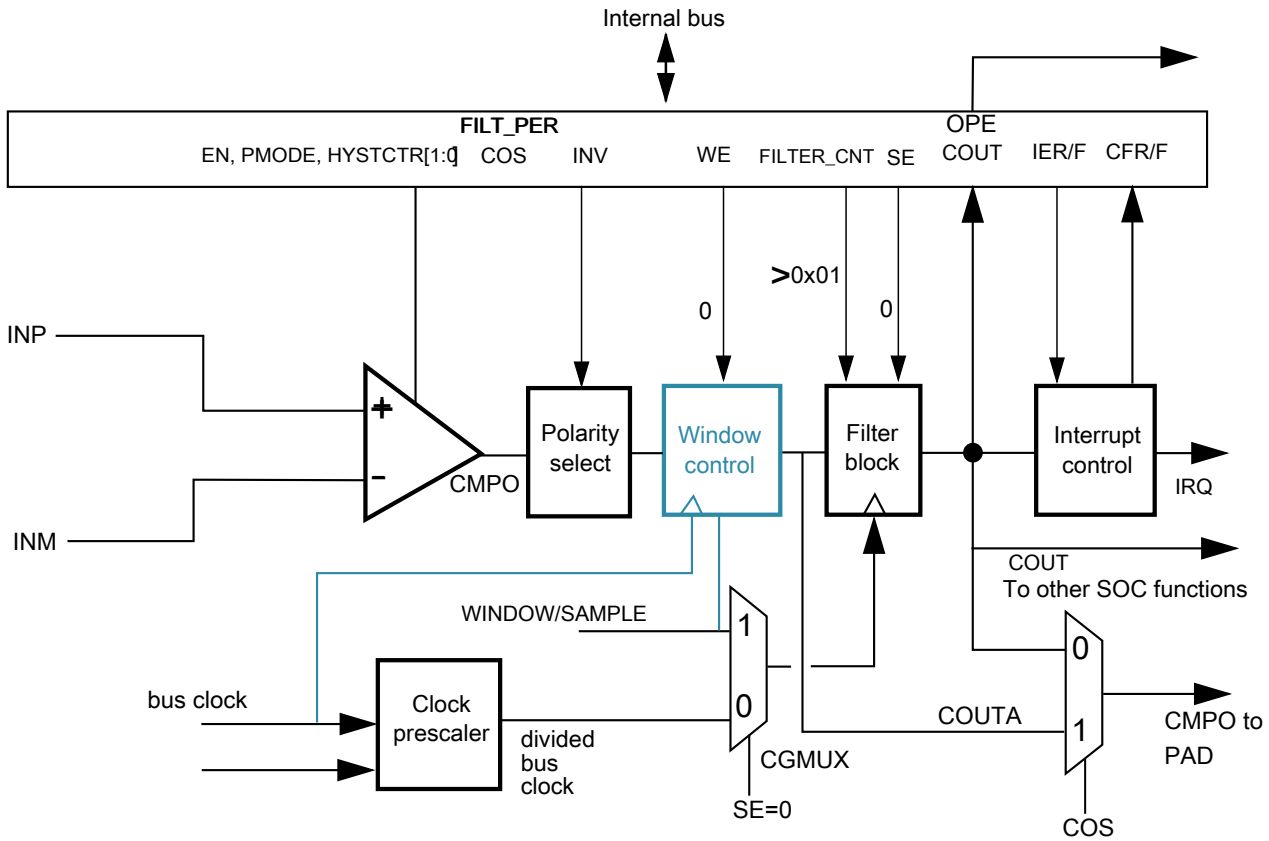


Figure 35-10. Sampled, Filtered (# 4B): sampling point internally derived

The only difference in operation between Sampled, Non-Filtered (# 3B) and Sampled, Filtered (# 4B) is that now, C0[FILTER_CNT]>1, which activates filter operation.

35.7.5 Windowed mode (#s 5A & 5B)

The following figure illustrates comparator operation in the Windowed mode, ignoring latency of the analog comparator, polarity select, and window control block. It also assumes that the polarity select is set to non-inverting state.

NOTE

The analog comparator output is passed to COUTA only when the WINDOW signal is high.

In actual operation, COUTA may lag the analog inputs by up to one bus clock cycle plus the combinational path delay through the comparator and polarity select logic.

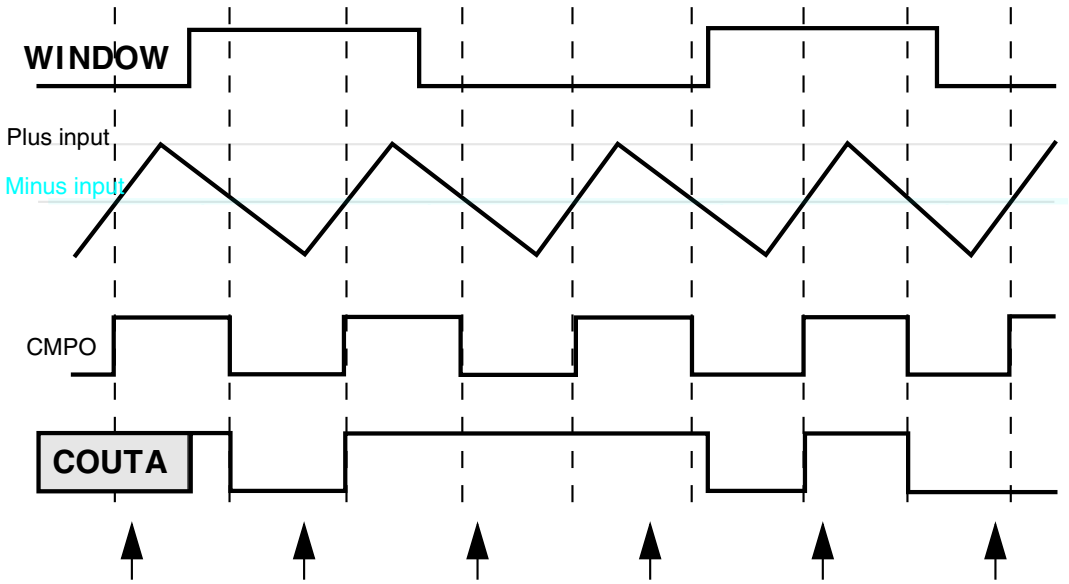


Figure 35-11. Windowed mode timing diagram

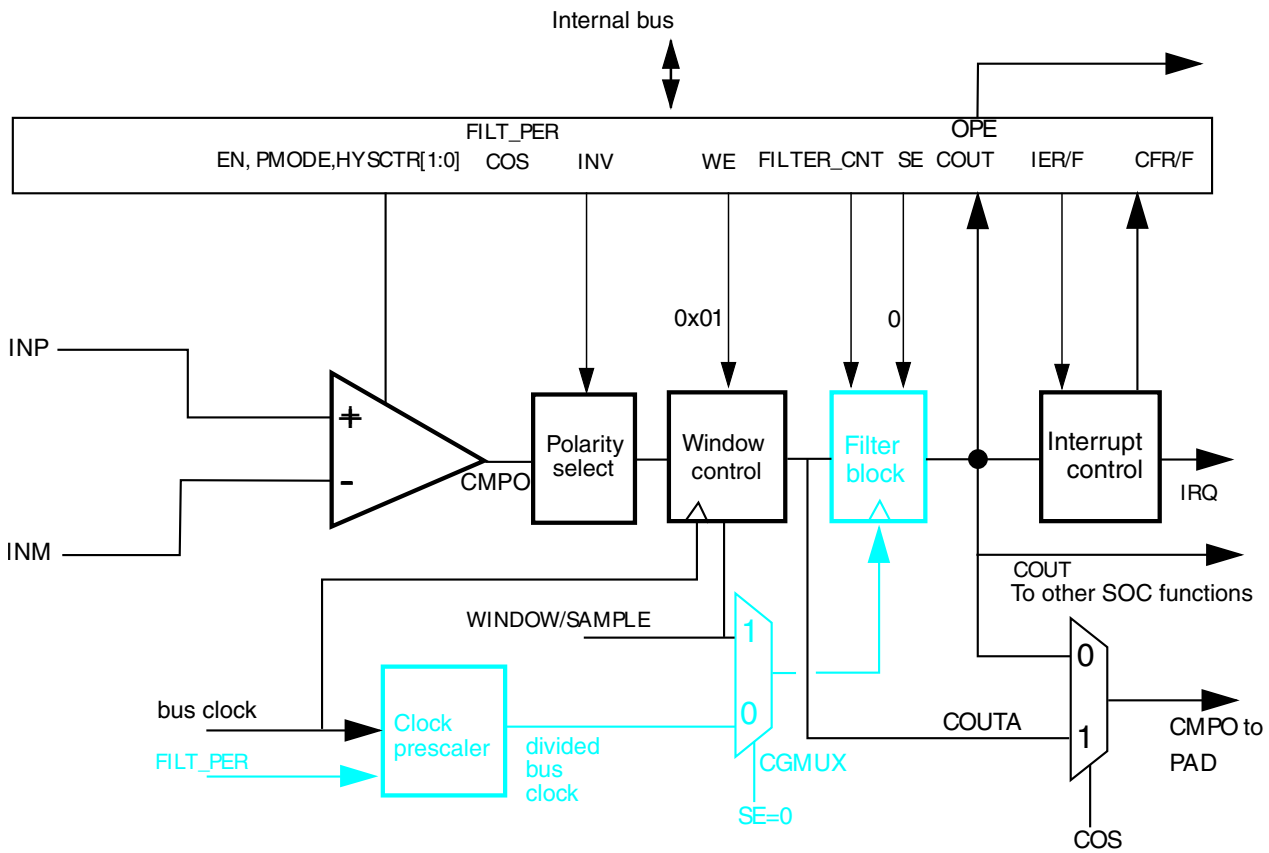


Figure 35-12. Windowed mode

For control configurations which result in disabling the filter block, see Figure 35-5.

When any windowed mode is active, COUTA is clocked by the bus clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

35.7.6 Windowed/Resampled mode (# 6)

The following figure uses the same input stimulus shown in Figure 35-11, and adds resampling of COUTA to generate COUT. Samples are taken at the time points indicated by the arrows in the figure. Again, prop delays and latency are ignored for the sake of clarity.

This example was generated solely to demonstrate operation of the comparator in windowed/resampled mode, and does not reflect any specific application. Depending upon the sampling rate and window placement, COUT may not see zero-crossing events detected by the analog comparator. Sampling period and/or window placement must be carefully considered for a given application.

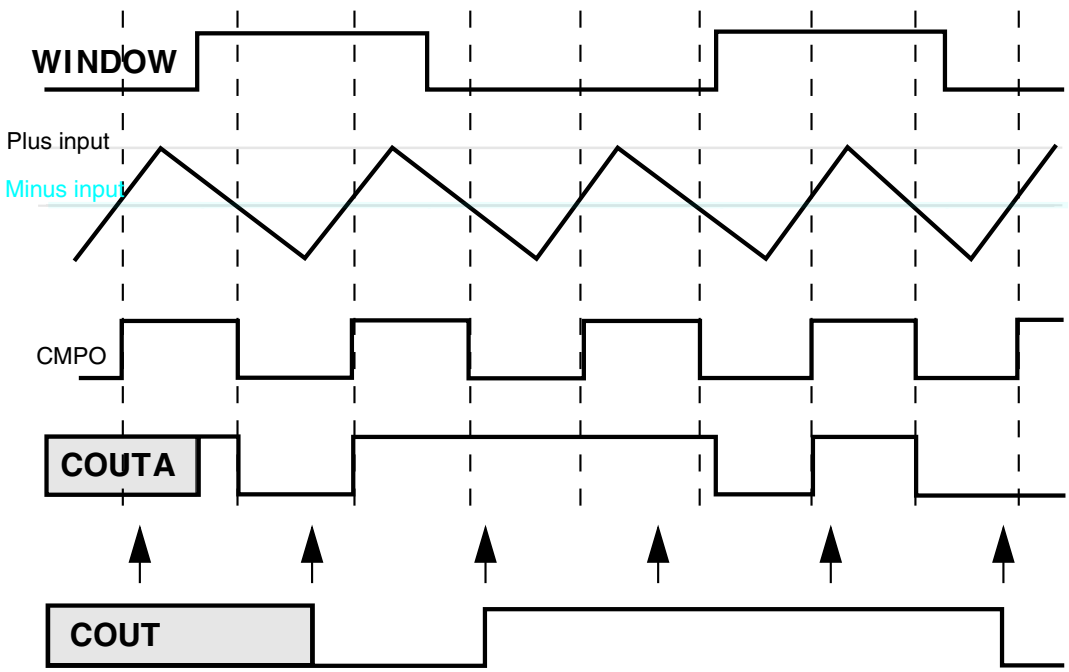


Figure 35-13. Windowed/resampled mode operation

This mode of operation results in an unfiltered string of comparator samples where the interval between the samples is determined by FPR[FILT_PER] and the bus clock rate. Configuration for this mode is virtually identical to that for the Windowed/Filtered Mode shown in the next section. The only difference is that the value of C0[FILTER_CNT] must be 1.

35.7.7 Windowed/Filtered mode (#7)

This is the most complex mode of operation for the comparator block, as it uses both windowing and filtering features. It also has the highest latency of any of the modes. This can be approximated: up to 1 bus clock synchronization in the window function + ((C0[FILTER_CNT] * C0[FPR]) + 1) * bus clock for the filter function.

When any windowed mode is active, COUTA is clocked by the bus clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

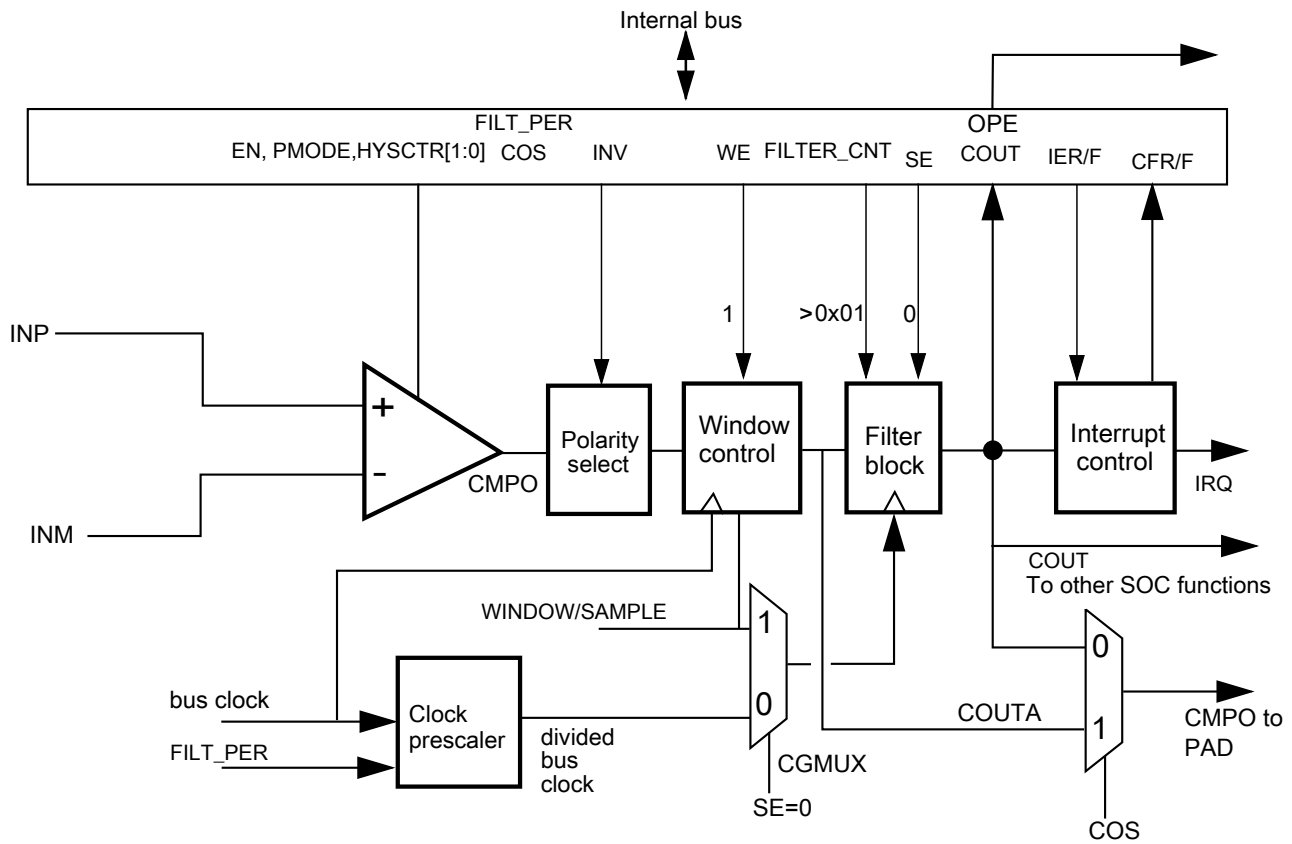


Figure 35-14. Windowed/Filtered mode

35.8 Memory map/register definitions

CMP memory map

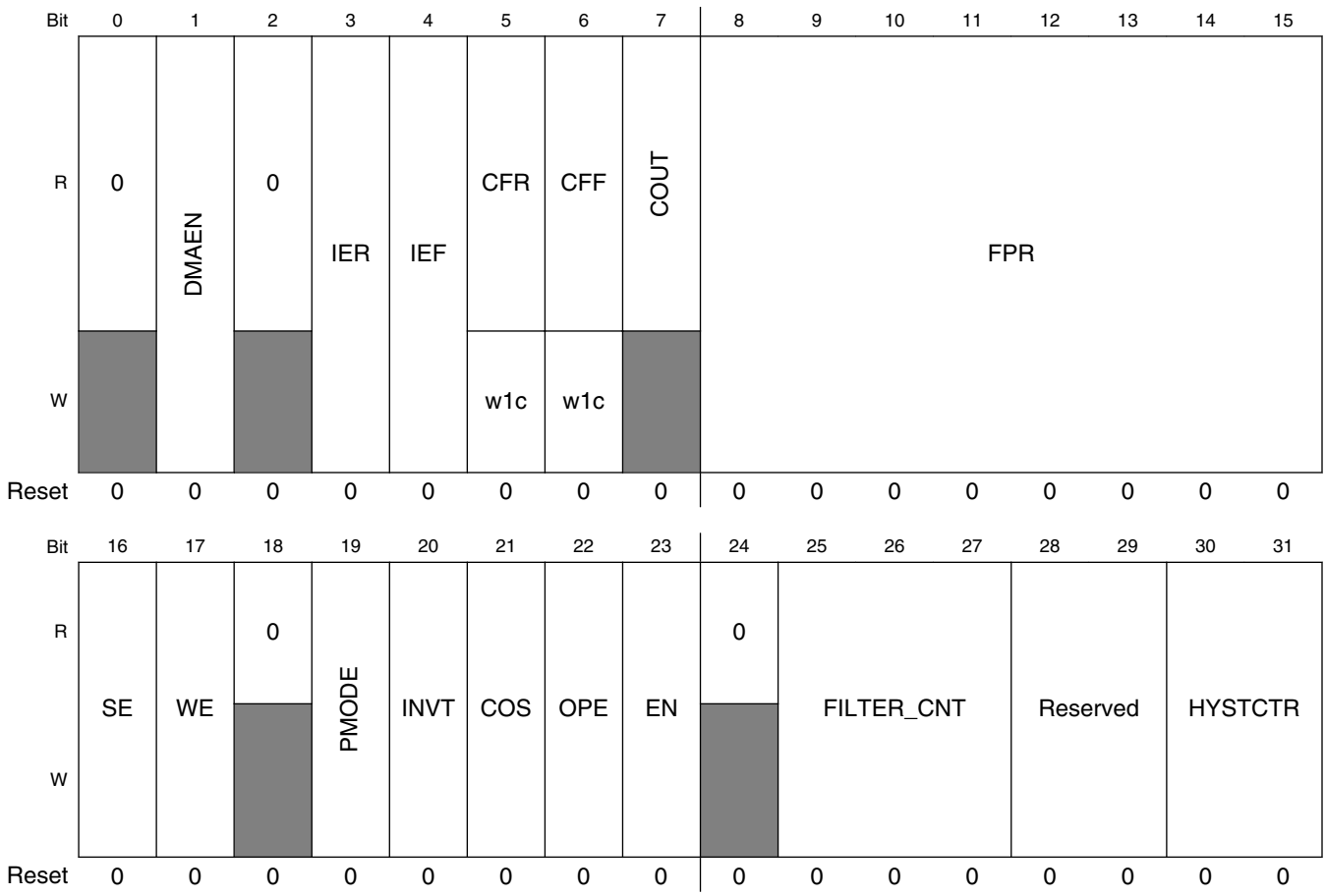
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	CMP Control Register 0 (CMP_C0)	32	R/W	0000_0000h	35.8.1/899
4	CMP Control Register 1 (CMP_C1)	32	R/W	0000_0000h	35.8.2/902
8	CMP Control Register 2 (CMP_C2)	32	R/W	0000_0000h	35.8.3/904

35.8.1 CMP Control Register 0 (CMP_C0)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 0h offset = 0h



CMP_C0 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DMAEN	DMA Enable Enables the DMA transfer triggered from the CMP module. When this field is set, a DMA request is asserted when CFR or CFF is set. 0 DMA is disabled. 1 DMA is enabled.

Table continues on the next page...

CMP_C0 field descriptions (continued)

Field	Description
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 IER	Comparator Interrupt Enable Rising Enables the CFR interrupt from the CMP. When this field is set, an interrupt will be asserted when CFR is set. 0 Interrupt is disabled. 1 Interrupt is enabled.
4 IEF	Comparator Interrupt Enable Falling Enables the CFF interrupt from the CMP. When this field is set, an interrupt will be asserted when CFF is set. 0 Interrupt is disabled. 1 Interrupt is enabled.
5 CFR	Analog Comparator Flag Rising Detects a rising-edge on COUT, when set, during normal operation. CFR is cleared by writing 1 to it. During Stop modes, CFR is level sensitive 0 A rising edge has not been detected on COUT. 1 A rising edge on COUT has occurred.
6 CFF	Analog Comparator Flag Falling Detects a falling-edge on COUT, when set, during normal operation. CFF is cleared by writing 1 to it. During Stop modes, CFF is level senstive . 0 A falling edge has not been detected on COUT. 1 A falling edge on COUT has occurred.
7 COUT	Analog Comparator Output Returns the current value of the Analog Comparator output, when read. The field is reset to 0 and will read as C0[INVT] when the Analog Comparator module is disabled, that is, when C0[EN] = 0. Writes to this field are ignored.
8–15 FPR	Filter Sample Period Specifies the sampling period, in bus clock cycles, of the comparator output filter, when C1[SE] = 0. Setting FPR to 0x0 disables the filter. Filter programming and latency details are provided in the CMP functional description. This field has no effect when C0[SE]= 1. In that case, the external SAMPLE signal is used to determine the sampling period.
16 SE	Sample Enable At any given time, either SE or WE can be set. If a write to this register attempts to set both, then SE is set and WE is cleared. However, avoid writing ones to both bit locations because this "11" case is reserved. 0 Sampling mode is not selected. 1 Sampling mode is selected.
17 WE	Windowing Enable At any given time, either SE or WE can be set. If a write to this register attempts to set both, then SE is set and WE is cleared. However, avoid writing ones to both bit locations because this "11" case is reserved.

Table continues on the next page...

CMP_C0 field descriptions (continued)

Field	Description
	0 Windowing mode is not selected. 1 Windowing mode is selected.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 PMODE	Power Mode Select 0 Low Speed (LS) comparison mode is selected. 1 High Speed (HS) comparison mode is selected.
20 INVT	Comparator invert This bit allows selecting the polarity of the analog comparator function. It is also driven to the COUT output (on both the device pin and as C0[COUT]) when CR1[OPE]=0. 0 Does not invert the comparator output. 1 Inverts the comparator output.
21 COS	Comparator Output Select 0 Set CMPO to equal COUT (filtered comparator output). 1 Set CMPO to equal COUTA (unfiltered comparator output).
22 OPE	Comparator Output Pin Enable The OPE bit enables the path from the comparator output to a selected pin. 0 When OPE is 0, the comparator output (after window/filter settings dependent on software configuration) is not available to a packaged pin. 1 When OPE is 1, and if the software has configured the comparator to own a packaged pin, the comparator is available in a packaged pin.
23 EN	Comparator Module Enable The EN bit enables the Analog Comparator Module. When the module is not enabled, the analog part remains in the off state, and consumes no power. When the same input is selected from analog mux to the positive and negative port, the comparator is disabled automatically. 0 Analog Comparator is disabled. 1 Analog Comparator is enabled.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 FILTER_CNT	Filter Sample Count This field specifies the number of consecutive samples that must agree prior to the comparator ouput filter accepting a new output state. For information regarding filter programming and latency, please see the Functional Description. 000 Filter is disabled. If SE = 1, then COUT is a logic zero (this is not a legal state, and is not recommended). If SE = 0, COUT = COUTA. 001 1 consecutive sample must agree (comparator output is simply sampled). 010 2 consecutive samples must agree. 011 3 consecutive samples must agree. 100 4 consecutive samples must agree. 101 5 consecutive samples must agree.

Table continues on the next page...

CMP_C0 field descriptions (continued)

Field	Description
	110 6 consecutive samples must agree. 111 7 consecutive samples must agree.
28–29 Reserved	This field is reserved.
30–31 HYSTCTR	Comparator hard block hysteresis control 00 The hard block output has no hysteresis internally. 01 The hard block output has 20 mv hysteresis internally. 10 The hard block output has 40 mv hysteresis internally. 11 The hard block output has 60 mv hysteresis internally.

35.8.2 CMP Control Register 1 (CMP_C1)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0							CHN7	CHN6	CHN5	CHN4	CHN3	CHN2	CHN1	CHN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DPSEL	DMSEL	PSEL			MSEL			DACEN	VRSEL	VOSEL					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CMP_C1 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 CHN7	Channel 7 input enable

Table continues on the next page...

CMP_C1 field descriptions (continued)

Field	Description
	Channel 7 of the input enable for the round-robin checker. If CHN7 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
9 CHN6	Channel 6 input enable Channel 6 of the input enable for the round-robin checker. If CHN6 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
10 CHN5	Channel 5 input enable Channel 5 of the input enable for the round-robin checker. If CHN5 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
11 CHN4	Channel 4 input enable Channel 4 of the input enable for the round-robin checker. If CHN4 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
12 CHN3	Channel 3 input enable Channel 3 of the input enable for the round-robin checker. If CHN3 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
13 CHN2	Channel 2 input enable Channel 2 of the input enable for the round-robin checker. If CHN2 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
14 CHN1	Channel 1 input enable Channel 1 of the input enable for the round-robin checker. If CHN1 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
15 CHN0	Channel 0 input enable Channel 0 of the input enable for the round-robin checker. If CHN0 is set, then the corresponding channel to the non-fixed mux port is enabled to check its voltage value in the round-robin mode. If the same channel is selected as the reference voltage, this bit has no effect.
16 DPSEL	DAC output to Plus side ANMUX selection 0 The analog mux_out (selected from the external inputs) is selected as the comparator plus side input. 1 The DAC out is selected as the comparator plus side input.
17 DMSEL	DAC output to Minus side ANMUX selection 0 The analog mux_out (selected from the external input) is selected as the comparator minus side input. 1 The DAC out is selected as the comparator input.
18–20 PSEL	Plus Input MUX Control Determines which input is selected for the plus input of the comparator. NOTE: When an inappropriate operation selects the same input for both MUXes, the comparator automatically shuts down to prevent itself from becoming a noise generator.

Table continues on the next page...

CMP_C1 field descriptions (continued)

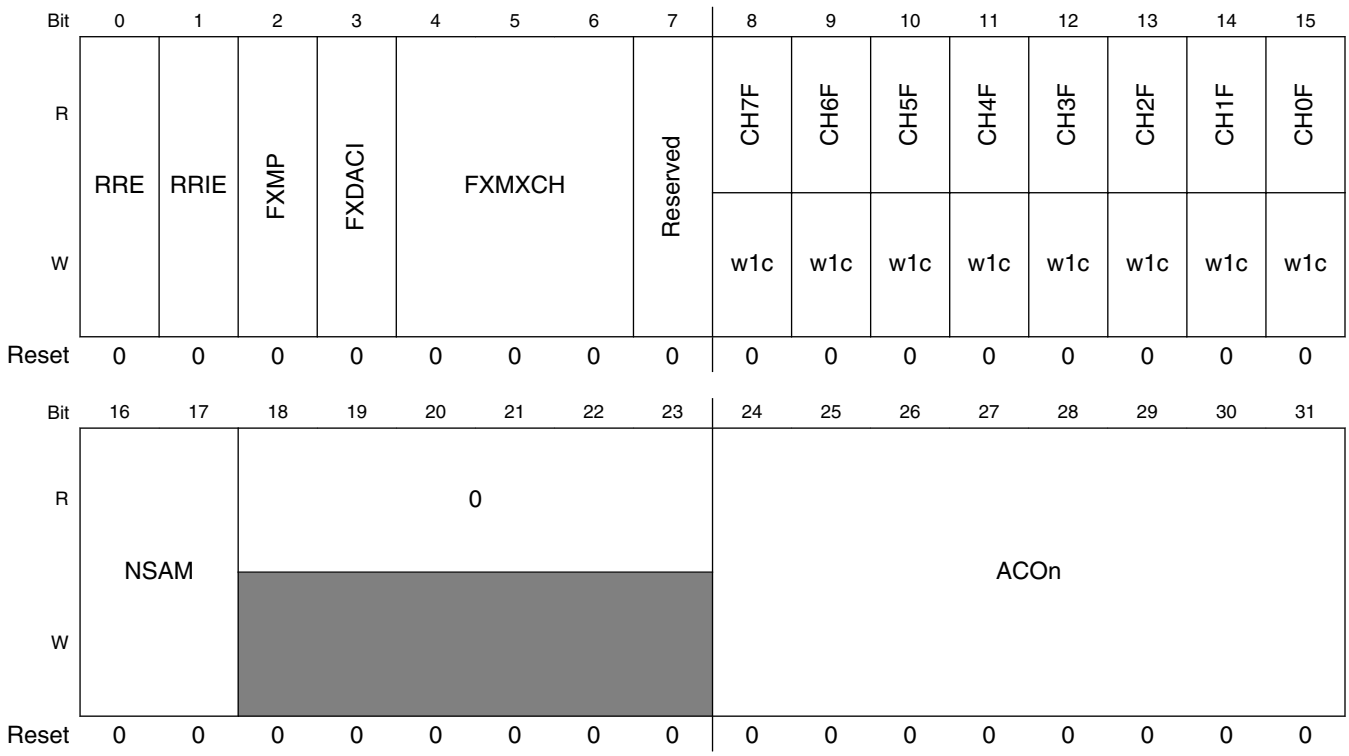
Field	Description
	Note: For the round robin mode of operation, the MSEL and PSEL bitfields in CMP*x*_C1 register must have different values. 000 IN0 001 IN1 010 IN2 011 IN3 100 IN4 101 IN5 110 IN6 111 IN7
21–23 MSEL	Minus Input MUX Control Determines which input is selected for the minus input of the comparator. NOTE: When an inappropriate operation selects the same input for both MUXes, the comparator automatically shuts down to prevent itself from becoming a noise generator. Note: For the round robin mode of operation, the MSEL and PSEL bitfields in CMP*x*_C1 register must have different values. 000 IN0 001 IN1 010 IN2 011 IN3 100 IN4 101 IN5 110 IN6 111 IN7
24 DACEN	DAC Enable This bit is used to enable the DAC. When the DAC is disabled, it is powered down to conserve power. 0 DAC is disabled. 1 DAC is enabled.
25 VRSEL	Supply Voltage Reference Source Select 0 Vin1 is selected as resistor ladder network supply reference Vin. 1 Vin2 is selected as resistor ladder network supply reference Vin.
26–31 VOSEL	DAC Output Voltage Select This bit selects an output voltage from one of 64 distinct levels. DACO = (Vin/64) * (VOSEL[5:0] + 1), so the DACO range is from Vin/64 to Vin.

35.8.3 CMP Control Register 2 (CMP_C2)

Access:

- Supervisor read/write
- User read/write

Address: 0h base + 8h offset = 8h



CMP_C2 field descriptions

Field	Description
0 RRE	Round-Robin Enable This bit enables the round-robin operation. 0 Round-robin operation is disabled. 1 Round-robin operation is enabled.
1 RRIE	Round-Robin interrupt enable This bit enables the interrupt/wake-up when the comparison result changes for a given channel. 0 The round-robin interrupt is disabled. 1 The round-robin interrupt is enabled when a comparison result changes from the last sample.
2 FXMP	Fixed MUX Port This bit is used to fix the analog mux port for the round-robin mode. 0 The Plus port is fixed. Only the inputs to the Minus port are swept in each round. 1 The Minus port is fixed. Only the inputs to the Plus port are swept in each round.
3 FXDACI	Fix DAC input to the fixed mux port side 0 The DAC output is not used as the reference input to the fixed mux port. 1 The DAC output is used as the reference input to the fixed mux port, regardless of the FXMXCH configuration.
4–6 FXMXCH	Fixed channel selection

Table continues on the next page...

CMP_C2 field descriptions (continued)

Field	Description
	This field indicates which channel in the mux port is fixed in a given round-robin mode. If FXDACI is set, FXMXCH has no effect. 000 Channel 0 is selected as the fixed reference input for the fixed mux port. 001 Channel 1 is selected as the fixed reference input for the fixed mux port. 010 Channel 2 is selected as the fixed reference input for the fixed mux port. 011 Channel 3 is selected as the fixed reference input for the fixed mux port. 100 Channel 4 is selected as the fixed reference input for the fixed mux port. 101 Channel 5 is selected as the fixed reference input for the fixed mux port. 110 Channel 6 is selected as the fixed reference input for the fixed mux port. 111 Channel 7 is selected as the fixed reference input for the fixed mux port.
7 Reserved	This field is reserved.
8 CH7F	Channel 7 input changed flag. This bit is set If the channel 7 input changed from the last comparison with the fixed mux port.
9 CH6F	Channel 6 input changed flag. This bit is set If the channel 6 input changed from the last comparison with the fixed mux port.
10 CH5F	Channel 5 input changed flag. This bit is set If the channel 5 input changed from the last comparison with the fixed mux port.
11 CH4F	Channel 4 input changed flag. This bit is set If the channel 4 input changed from the last comparison with the fixed mux port.
12 CH3F	Channel 3 input changed flag. This bit is set If the channel 3 input changed from the last comparison with the fixed mux port.
13 CH2F	Channel 2 input changed flag. This bit is set If the channel 2 input changed from the last comparison with the fixed mux port.
14 CH1F	Channel 1 input changed flag. This bit is set If the channel 1 input changed from the last comparison with the fixed mux port.
15 CH0F	Channel 0 input changed flag. This bit is set If the channel 0 input changed from the last comparison with the fixed mux port.
16–17 NSAM	Number of sample clocks For a given channel, this field specifies how many round-robin clock cycles later the sample takes place. 00 The comparison result is sampled in . 01 The sampling takes place 1 round-robin clock cycle after the next cycle of the round-robin clock. 10 The sampling takes place 2 round-robin clock cycles after the next cycle of the round-robin clock. 11 The sampling takes place 3 round-robin clock cycles after the next cycle of the round-robin clock.
18–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 ACOn	The result of the input comparison for channel *n* . This field stores the latest comparison result of the input channel *n* with the fixed mux port. Reading this bit returns the latest comparison result. Writing this field defines the pre-set state of channel *n*.

35.9 CMP functional description

The CMP module can be used to compare two analog input voltages applied to INP and INM. CMPO is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. This signal can be selectively inverted by setting C0[INVT] = 1.

C0[IER] and C0[IEF] are used to select the condition that causes the CMP module to assert an interrupt to the processor. C0[CFF] is set on a falling edge, and C0[CFR] is set on a rising edge of the comparator output. The optionally filtered CMPO can be read directly through C0[COUT].

35.9.1 Initialization

A typical startup sequence is as follows.

The time required to stabilize COUT is the power-on delay of the comparators plus the largest propagation delay from a selected analog source through the analog comparator, windowing function, and filter. See the datasheet for power-on delays of the comparators. The windowing function has a maximum of one bus clock period delay. The filter delay is specified in the Low-pass filter section.

During operation, the propagation delay of the selected data paths must always be considered. It may take many bus clock cycles for COUT and C0[CFR]/C0[CFF] to reflect an input change or a configuration change to one of the components involved in the data path.

When programmed for filtering modes, COUT initially equals 0 until sufficient clock cycles have elapsed to fill all stages of the filter. This occurs even if COUTA is at a logic 1.

35.9.2 Low-pass filter

The low-pass filter operates on the unfiltered and unsynchronized and optionally inverted comparator output COUTA and generates the filtered and synchronized output COUT. Both COUTA and COUT can be configured as module outputs and are used for different purposes within the system.

Synchronization and edge detection are always used to determine status register bit values. They also apply to COUT for all sampling and windowed modes. Filtering can be performed using an internal timebase defined by FPR[FILT_PER], or using an external SAMPLE input to determine sample time.

The need for digital filtering and the amount of filtering is dependent on user requirements. Filtering can become more useful in the absence of an external hysteresis circuit. Without external hysteresis, high-frequency oscillations can be generated at COUTA when the selected INM and INP input voltages differ by less than the offset voltage of the differential comparator.

35.9.2.1 Enabling filter modes

Filter modes can be enabled by:

- Setting C0[FILTER_CNT] > 0x01 and
- Setting C0[FPR] to a nonzero value or setting C0[SE]=1

If using the divided bus clock to drive the filter, it samples COUTA every C0[FPR] bus clock cycles.

The filter output is at logic 0 when first initalized, and subsequently changes when all the consecutive C0[FILTER_CNT] samples agree that the output value has changed. In other words, C0[COUT] is 0 for some initial period, even when COUTA is at logic 1.

Setting both C0[SE] and C0[FPR] to 0 disables the filter and eliminates switching current associated with the filtering process.

Note

Always switch to this setting prior to making any changes in filter parameters. This resets the filter to a known state. Switching C0[FILTER_CNT] on the fly without this intermediate step can result in unexpected behavior.

If C0[SE]=1, the filter samples COUTA on each positive transition of the sample input. The output state of the filter changes when all the consecutive C0[FILTER_CNT] samples agree that the output value has changed.

35.9.2.2 Latency issues

The value of C0[FPR] or SAMPLE period must be set such that the sampling period is just longer than the period of the expected noise. This way a noise spike will corrupt only one sample. The value of C0[FILTER_CNT] must be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of C0[FILTER_CNT].

The values of C0[FPR] or SAMPLE period and C0[FILTER_CNT] must also be traded off against the desire for minimal latency in recognizing actual comparator output transitions. The probability of detecting an actual output change within the nominal latency is the probability of a correct sample raised to the power of C0[FILTER_CNT].

The following table summarizes maximum latency values for the various modes of operation *in the absence of noise*. Filtering latency is restarted each time an actual output transition is masked by noise.

Table 35-6. Comparator sample/filter maximum latencies

Mode #	C0[EN]	C0[WE]	C0[SE]	C0[FILTER_CNT]	Co[FPR]	Operation	Maximum latency[1]
1	0	X	X	X	X	Disabled	N/A
2A	1	0	0	0x00	X	Continuous Mode	T_{PD}
2B	1	0	0	X	0x00		
3A	1	0	1	0x01	X	Sampled, Non-Filtered mode	$T_{PD} + T_{SAMPLE} + T_{per}$
3B	1	0	0	0x01	> 0x00		$T_{PD} + (C0[FPR] * T_{per}) + T_{per}$
4A	1	0	1	> 0x01	X	Sampled, Filtered mode	$T_{PD} + (C0[FILTER_CNT] * T_{SAMPLE}) + T_{per}$
4B	1	0	0	> 0x01	> 0x00		$T_{PD} + (C0[FILTER_CNT] * C0[FPR] \times T_{per}) + T_{per}$
5A	1	1	0	0x00	X	Windowed mode	$T_{PD} + T_{per}$
5B	1	1	0	X	0x00		$T_{PD} + T_{per}$
6	1	1	0	0x01	0x01 - 0xFF	Windowed / Resampled mode	$T_{PD} + (C0[FPR] * T_{per}) + 2T_{per}$
7	1	1	0	> 0x01	0x01 - 0xFF	Windowed / Filtered mode	$T_{PD} + (C0[FILTER_CNT] * C0[FPR] \times T_{per}) + 2T_{per}$

1. T_{PD} represents the intrinsic delay of the analog component plus the polarity select logic. T_{SAMPLE} is the clock period of the external sample clock. T_{per} is the period of the bus clock.

35.10 Interrupts

The CMP module is capable of generating an interrupt on either the rising- or falling-edge of the comparator output, or both. Assuming the CMP DMA enable bit is not set, the following table gives the conditions in which the interrupt request is asserted and deasserted.

Table 35-7. CMP interrupt generations

When	Then
C0[IER] and C0[CFR] are set	The interrupt request is asserted
C0[IEF] and C0[CFF] are set	The interrupt request is asserted
C0[IER] and C0[CFR] are cleared for a rising-edge interrupt	The interrupt request is deasserted
C0[IEF] and C0[CFF] are cleared for a falling-edge interrupt	The interrupt request is deasserted

35.11 DMA support

Normally, the CMP generates a CPU interrupt if there is a change on the COUT. When DMA support is enabled by setting C0[DMAEN] and the interrupt is enabled by setting C0[IER], C0[IEF], or both, the corresponding change on COUT forces a DMA transfer request rather than a CPU interrupt instead. When the DMA has completed the transfer, it sends a transfer completing indicator signal that deasserts the DMA transfer request and clears the flag to allow a subsequent change on comparator output to occur and force another DMA request.

35.12 DAC functional description

This section provides DAC functional description.

35.12.1 Digital-to-analog converter block diagram

The following figure shows the block diagram of the DAC module. It contains a 64-tap resistor ladder network and a 64-to-1 multiplexer, which selects an output voltage from one of 64 distinct levels that outputs from DACO. It is controlled through the DAC

Control Register (DACCR). Its supply reference source can be selected from two sources V_{in1} and V_{in2}. The module can be powered down or disabled when not in use. When in Disabled mode, DACO is connected to the analog ground.

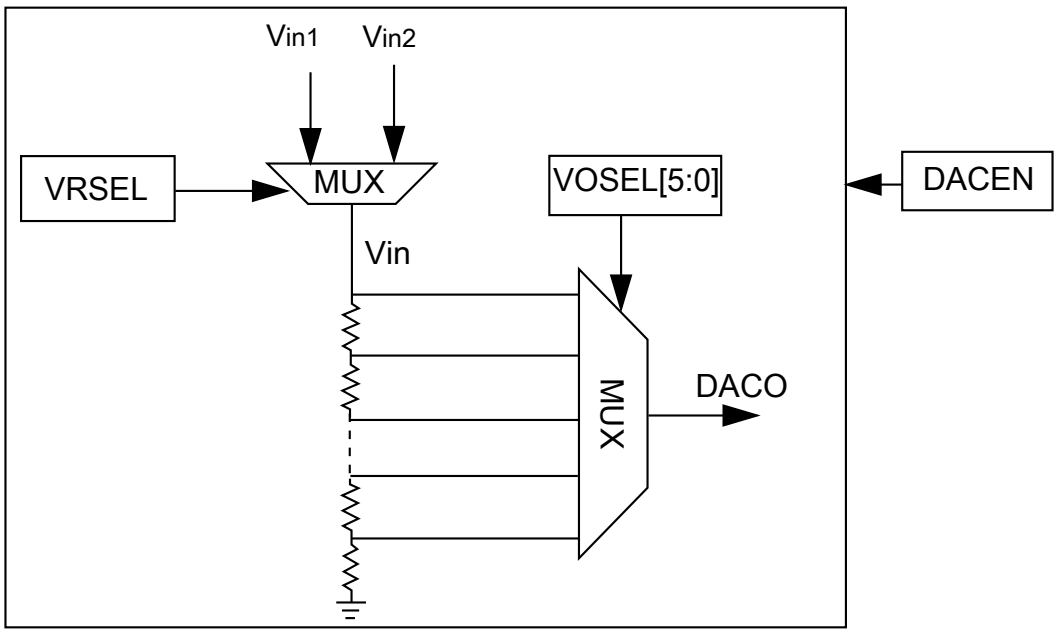


Figure 35-15. 6-bit DAC block diagram

35.12.2 Voltage reference source select

- V_{in1} must be used to connect to the primary voltage source as supply reference of 64 tap resistor ladder
- V_{in2} must be used to connect to an alternate voltage source, or primary source, if an alternate voltage source is not available

35.13 DAC resets

This module has a single reset input, corresponding to the chip-wide peripheral reset.

35.14 DAC clocks

This module has a single clock input, the bus clock.

35.15 DAC interrupts

This module has no interrupts.

35.16 Trigger mode

The CMP and the 6-bit DAC are designed to support the trigger mode operation, which is enabled when the MCU enters STOP modes with CMP_*x*_C2[RRE] set.

With this mode enabled, the trigger events that include the operation clock and a trigger start signal will initiate a compare sequence that must first enable the CMP and DAC prior to performing a CMP operation and capturing the output. A fixed channel for either the plus side mux or the minus side mux is selected by software with CMP_*x*_C2[FXMP] and CMP_*x*_C2[FXMXCH].

The active channels selected by CMP_*x*_C1[CHN*n*] are then routed to the non-fixed channel mux and compared with the reference input in a round-robin manner. In order to meet the comparator stabilization time, after the configurable number of operation clocks defined by CMP_*x*_C2[NSAM], the comparison result is sampled for the selected channel. A software pre-progammed state for each channel is configured by writing to CMP_*x*_C2[ACO*n*] field. After all the active channels are sampled, if the comparison result changes from its pre-programmed state, the corresponding flag in CMP_*x*_C2[CH*n*F] is set. If CMP_*x*_C2[RRIE] is set, an asynchronous reset is asserted to bring the MCU out of STOP mode.

Note that these flags do not support generating a DMA transfer event.

This mode is active when the MCU is in STOP mode, so none of the window/filter functions are available. A basic assumption of this mode is that the selected inputs are changing at a much slower rate than the operation clock. It is suggested to configure the comparator in low power comparison mode as well.

The following diagram shows the basic flow of this mode. In the diagram, CMP_*x*_C1[CHN1], CMP_*x*_C1[CHN3], and CMP_*x*_C1[CHN7] are set, so channels #1, #3, and #7 are selected for round robin. CMP_*x*_C2[NSAM] is set to 2'b01, so one clock later the comparison result of the selected channel is sampled. When channel #7 is compared, the result is sampled, and round robin ends. If any of the comparison results from channel #1, #3, or #7 changed from their programmed value (written to CMP_*x*_C2[ACO1], CMP_*x*_C2[ACO3], and CMP_*x*_C2[ACO7]), an interrupt is generated to wake up the MCU from the STOP mode. Software can then poll the CMP_*x*_C2[CH*n*F] to see which channel input(s) changed value during the STOP mode.

NOTE

In round robin mode, it should be ensured that the RTC_CLK period is greater than the comparison time corresponding to the value of CMPx_C0[PMODE].

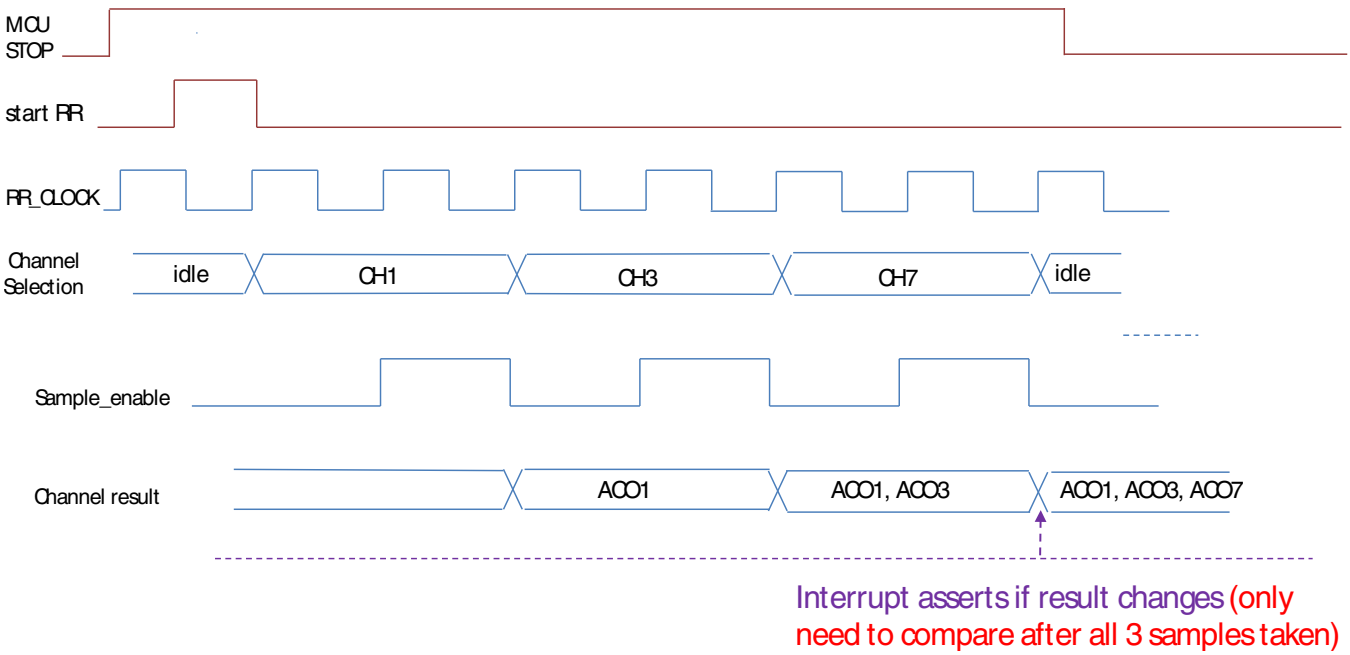


Figure 35-16. Trigger mode

Chapter 36
Body Cross triggering unit (BCTU)

36.1 Chip-specific BCTU information

36.1.1 BCTU configuration

The enhanced BCTU allows synchronization between an ADC conversion and a timer event. All channels from the 2x eMIOS modules, and 3 PIT channels can be used to trigger an ADC conversion. A single event (eMIOS channel or PIT) can trigger a group of ADC channels to be converted.

36.1.2 Trigger sources

The BCTU has input triggers from all eMIOS channels, a number of PIT channels. The BCTU can be triggered from the software also. The following table lists the BCTU trigger sources.

Table 36-1. BCTU Trigger sources

BCTU Trigger Number	Module	Source
0	eMIOS_0	Channel_0
1	eMIOS_0	Channel_1
2	eMIOS_0	Channel_2
3	eMIOS_0	Channel_3
4	eMIOS_0	Channel_4
5	eMIOS_0	Channel_5
6	eMIOS_0	Channel_6
7	eMIOS_0	Channel_7
8	eMIOS_0	Channel_8
9	eMIOS_0	Channel_9
10	eMIOS_0	Channel_10

Table continues on the next page...

Table 36-1. BCTU Trigger sources (continued)

BCTU Trigger Number	Module	Source
11	eMIOS_0	Channel_11
12	eMIOS_0	Channel_12
13	eMIOS_0	Channel_13
14	eMIOS_0	Channel_14
15	eMIOS_0	Channel_15
16	eMIOS_0	Channel_16
17	eMIOS_0	Channel_17
18	eMIOS_0	Channel_18
19	eMIOS_0	Channel_19
20	eMIOS_0	Channel_20
21	eMIOS_0	Channel_21
22	eMIOS_0	Channel_22
23	PIT_RTI	PIT_3
24	eMIOS_0	Channel_24
25	eMIOS_0	Channel_25
26	eMIOS_0	Channel_26
27	eMIOS_0	Channel_27
28	eMIOS_0	Channel_28
29	eMIOS_0	Channel_29
30	eMIOS_0	Channel_30
31	eMIOS_0	Channel_31
32	eMIOS_1	Channel_0
33	eMIOS_1	Channel_1
34	eMIOS_1	Channel_2
35	eMIOS_1	Channel_3
36	eMIOS_1	Channel_4
37	eMIOS_1	Channel_5
38	eMIOS_1	Channel_6
39	eMIOS_1	Channel_7
40	eMIOS_1	Channel_8
41	eMIOS_1	Channel_9
42	eMIOS_1	Channel_10
42	eMIOS_1	Channel_11
44	eMIOS_1	Channel_12
45	eMIOS_1	Channel_13
46	eMIOS_1	Channel_14
47	eMIOS_1	Channel_15
48	eMIOS_1	Channel_16
49	eMIOS_1	Channel_17

Table continues on the next page...

Table 36-1. BCTU Trigger sources (continued)

BCTU Trigger Number	Module	Source
50	eMIOS_1	Channel_18
51	eMIOS_1	Channel_19
52	eMIOS_1	Channel_20
53	eMIOS_1	Channel_21
54	eMIOS_1	Channel_22
55	PIT_RTI	PIT_7
56	eMIOS_1	Channel_24
57	eMIOS_1	Channel_25
58	eMIOS_1	Channel_26
59	eMIOS_1	Channel_27
60	eMIOS_1	Channel_28
61	eMIOS_1	Channel_29
62	eMIOS_1	Channel_30
63	eMIOS_1	Channel_31

36.1.3 BCTU's stop-stop ack protocol

The BCTU module provides the acknowledgement to the stop request:

- Immediately if there is no on-going ADC conversion requested by BCTU.
- After completion of ADC conversion, if there is any on-going ADC conversion requested by BCTU.
- The acknowledgement signal is provided as long as the request signal to the IP is asserted.

36.2 Introduction

The Body Cross-triggering Unit (BCTU) is a module that is used to trigger the on chip ADCs. The trigger sources are generated by on chip timer modules. Usually in Body domain of applications the timers used are the enhanced Modular IO Sub-system (eMIOS) and the Programmable Interrupt Timer (PIT). In addition to triggering ADC single conversions, the BCTU also implements a LIST—a set of registers programmed by software that stores ADC channel numbers—that triggers several conversions based on a single trigger occurrence. Up to 64 trigger inputs are available. The CPU can also be a source of conversion triggers, thus implementing a software trigger control. Up to 2 ADC modules are supported. The conversion results are stored in a register which is available for CPU read access. Figure below shows the BCTU top level diagram.

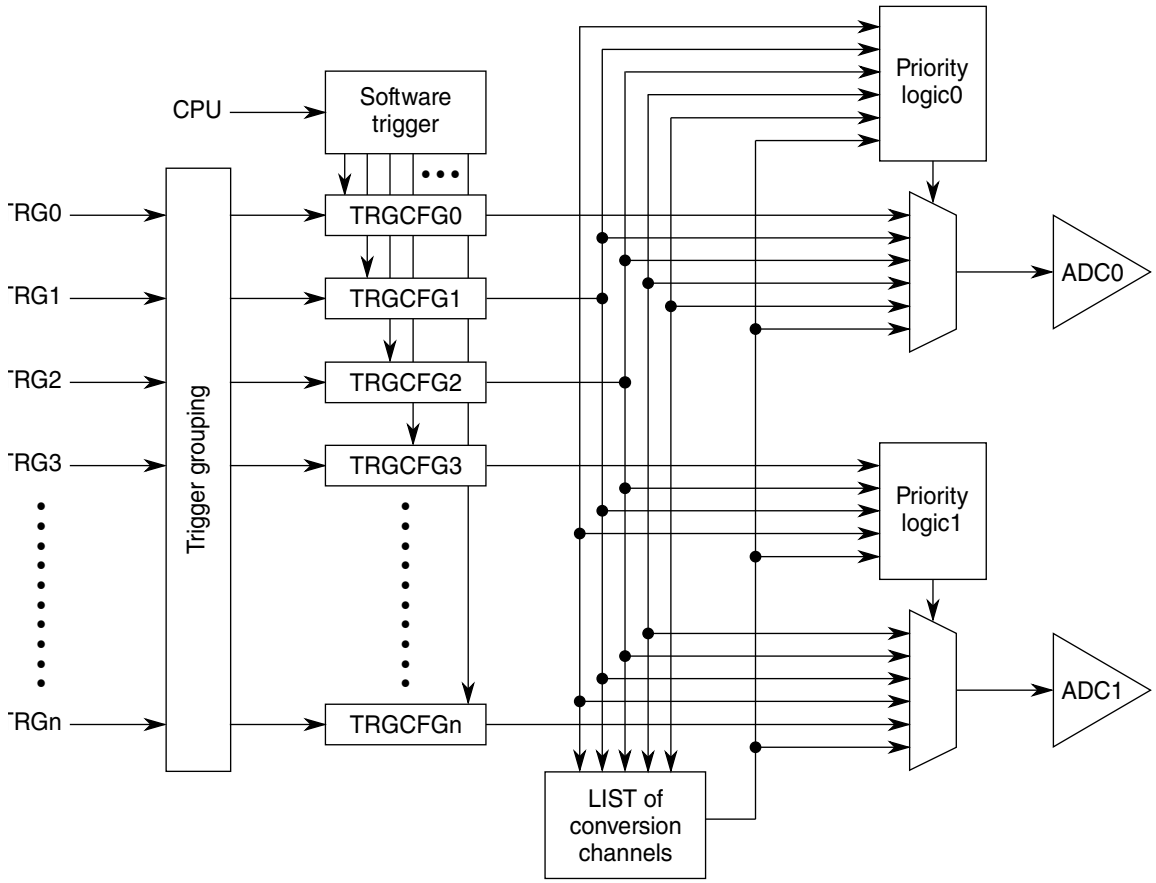


Figure 36-1. BCTU block diagram

36.3 BCTU overview

The BCTU receives several triggers from on chip timers and a software trigger controlled by the host CPU. These signals are used in order to generate the ADC trigger signals for up to 2 ADC modules. One trigger input selects one trigger configuration register. The trigger configuration register has the information about the target ADC and channel for the conversion. Alternatively a list of ADC channels can be associated with a trigger configuration register. In this case the target ADC converts a sequence of channels which are stored in the LIST.

The BCTU inputs are up to 64 trigger signals, which are level-sensitive; thus, if a logical level 1 is detected, an event is captured by the BCTU.

The figure above shows a high level BCTU block diagram. The BCTU hardware architecture consists of:

- Up to 64 conversion configuration registers TRGCFG
- One priority selection logic per ADC

- Up to 4 selection multiplexers to select one trigger at a time for each ADC
- A LIST of ADC channels

The priority logic is responsible for selecting among all active trigger inputs the one which is used to trigger an ADC conversion. The basic operation of the priority logic is to select the smallest trigger number as the higher priority. Thus if TRG[0] and TRG[1] are active TRG[0] is selected first and then TRG[1]. This priority scheme is used only if TRG[0] and TRG[1] target the same ADC. If a different ADC is selected by each trigger then there is no conflict and both triggers are selected at the same time. In addition to the trigger priority scheme the BCTU provides a LIST of channels that are used in a sequence of conversions. The LIST is a set of registers programmed by the software that stores ADC Channel numbers. The LIST also stores information of the last conversion to be executed in a sequence of conversions. A regular single conversion or a LIST of conversions are configured in the Trigger Configuration register. Since the ADC is selected by the ADC_SEL bit in the TRGCFG register for one single or a list of conversions, the LIST does not store the target ADC selection, thus all channels in the LIST refers to the ADCs specified in the TRGCFG register.

NOTE

Differently from the single command, the LIST can execute conversions on more than one ADC. Up to two ADCs can be addressed in a LIST.

36.4 BCTU Interacting with other peripherals

Figure below shows an example of how the BCTU interacts with on chip peripherals.

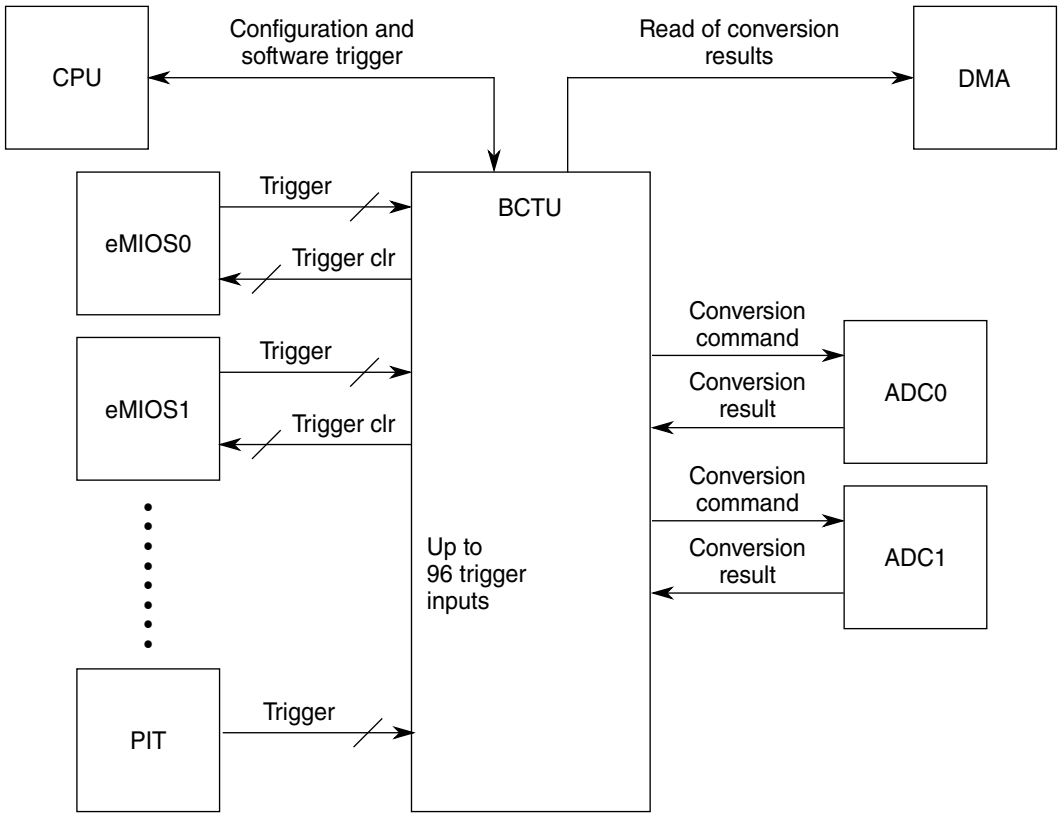


Figure 36-2. BCTU connections to other on chip modules

The timers (i.e. eMIOS and PIT) generate triggers which are asserted at a specific time when the ADC conversion should be done. The transition, from zero to one, of these signals is captured by the BCTU logic and stored until the corresponding ADC conversion starts. At this time the BCTU asserts a trigger clear signal which clears both this internal captured trigger signal and the external trigger source if they are still asserted.

NOTE

Since the timer trigger may remain active until the corresponding conversion has initiated, there could be a case where another match occurs in the timer (i.e. eMIOS or PIT). The detection of that condition is not done by the BCTU logic. It is up to the application software to evaluate such a trigger overrun.

36.5 BCTU Modes of operation

The BCTU has following modes of operation, which are described below.

- Configuration mode

- Normal mode
- Low power mode

36.5.1 Configuration Mode

In the configuration mode the BCTU does not issue triggers to the ADCs connected to it. This mode allows for the software to program the TRGCFG registers, freely, without generating any spurious trigger to the ADCs. All trigger inputs are disabled. The exit from the configuration mode is made by setting the global enable bit GTRGEN in the MCR register. This procedure allows to enable all trigger inputs, which are already individually enabled by the TRIGEN bit in the TRGCFG register, with a single software access.

NOTE

In order to reinitialize the trigger input interface registers, the application software should clear all triggers by writing 1 to the TRG_FLAG bit in all TRGCFG registers before asserting the GTRGEN bit . This procedure clears the BCTU trigger input registers as well as the trigger sources that are connected to BCTU trigger clear signals (i.e. eMIOS).

36.5.2 Normal Mode

In the normal mode of operation the BCTU generates triggers for the ADCs based on trigger inputs from the on-chip timer modules. Due to backward compatibility reasons this is the mode selected coming out of reset.

36.5.3 Low Power Mode

The low power mode is entered when the MDIS bit in the MCR register is set. In this mode the BCTU is disabled, meaning that all trigger inputs are disregarded.

No triggers are issued to the ADCs connected to it. If MDIS is set any on going conversion is finished before the low power mode is entered. The TRG and NDATAx flags are not set when the module enters in low power mode.

If there are pending active triggers, they remain active (no trigger clear signal is issued). In this case it is up for the application software to clear the pending triggers, if required.

In the low power mode, the clocks for the BCTU module may be stopped, thus allowing to reduce the dynamic power consumption of the module. The exit from low power mode is made by clearing the MDIS bit in the MCR register. The module becomes immediately active after the MDIS bit is cleared, meaning that if the GTRGEN bit is set any active trigger input is processed and the corresponding ADC is triggered in order to execute a new conversion. In low power mode the ADCDRx registers are available for software read operation.

36.6 Memory Map

BCTU registers are implemented in 32-bit. 8-bit write operations are allowed. The coherency in this case must be provided by the application software since there is no coherence mechanism built into the BCTU hardware.

NOTE

RESERVED bit registers, when available for write operations, should be written always with logic ‘0’. Read access does not return meaningful data from RESERVED bit registers.

NOTE

Registers with write protection, if protection mechanism is active and an attempt for writing to those registers is done, will not cause any change to the register content and a transfer error will be issued.

BCTU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (BCTU_MCR)	32	R/W	0000_0000h	36.6.1/925
8	Module Status register (BCTU_MSR)	32	R/W	0000_0000h	36.6.2/927
18	Trigger Configuration Register (BCTU_TRGCFG_0)	32	R/W	0000_0000h	36.6.3/931
1C	Trigger Configuration Register (BCTU_TRGCFG_1)	32	R/W	0000_0000h	36.6.3/931
20	Trigger Configuration Register (BCTU_TRGCFG_2)	32	R/W	0000_0000h	36.6.3/931
24	Trigger Configuration Register (BCTU_TRGCFG_3)	32	R/W	0000_0000h	36.6.3/931
28	Trigger Configuration Register (BCTU_TRGCFG_4)	32	R/W	0000_0000h	36.6.3/931
2C	Trigger Configuration Register (BCTU_TRGCFG_5)	32	R/W	0000_0000h	36.6.3/931
30	Trigger Configuration Register (BCTU_TRGCFG_6)	32	R/W	0000_0000h	36.6.3/931
34	Trigger Configuration Register (BCTU_TRGCFG_7)	32	R/W	0000_0000h	36.6.3/931
38	Trigger Configuration Register (BCTU_TRGCFG_8)	32	R/W	0000_0000h	36.6.3/931
3C	Trigger Configuration Register (BCTU_TRGCFG_9)	32	R/W	0000_0000h	36.6.3/931
40	Trigger Configuration Register (BCTU_TRGCFG_10)	32	R/W	0000_0000h	36.6.3/931

Table continues on the next page...

BCTU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
44	Trigger Configuration Register (BCTU_TRGCFG_11)	32	R/W	0000_0000h	36.6.3/931
48	Trigger Configuration Register (BCTU_TRGCFG_12)	32	R/W	0000_0000h	36.6.3/931
4C	Trigger Configuration Register (BCTU_TRGCFG_13)	32	R/W	0000_0000h	36.6.3/931
50	Trigger Configuration Register (BCTU_TRGCFG_14)	32	R/W	0000_0000h	36.6.3/931
54	Trigger Configuration Register (BCTU_TRGCFG_15)	32	R/W	0000_0000h	36.6.3/931
58	Trigger Configuration Register (BCTU_TRGCFG_16)	32	R/W	0000_0000h	36.6.3/931
5C	Trigger Configuration Register (BCTU_TRGCFG_17)	32	R/W	0000_0000h	36.6.3/931
60	Trigger Configuration Register (BCTU_TRGCFG_18)	32	R/W	0000_0000h	36.6.3/931
64	Trigger Configuration Register (BCTU_TRGCFG_19)	32	R/W	0000_0000h	36.6.3/931
68	Trigger Configuration Register (BCTU_TRGCFG_20)	32	R/W	0000_0000h	36.6.3/931
6C	Trigger Configuration Register (BCTU_TRGCFG_21)	32	R/W	0000_0000h	36.6.3/931
70	Trigger Configuration Register (BCTU_TRGCFG_22)	32	R/W	0000_0000h	36.6.3/931
74	Trigger Configuration Register (BCTU_TRGCFG_23)	32	R/W	0000_0000h	36.6.3/931
78	Trigger Configuration Register (BCTU_TRGCFG_24)	32	R/W	0000_0000h	36.6.3/931
7C	Trigger Configuration Register (BCTU_TRGCFG_25)	32	R/W	0000_0000h	36.6.3/931
80	Trigger Configuration Register (BCTU_TRGCFG_26)	32	R/W	0000_0000h	36.6.3/931
84	Trigger Configuration Register (BCTU_TRGCFG_27)	32	R/W	0000_0000h	36.6.3/931
88	Trigger Configuration Register (BCTU_TRGCFG_28)	32	R/W	0000_0000h	36.6.3/931
8C	Trigger Configuration Register (BCTU_TRGCFG_29)	32	R/W	0000_0000h	36.6.3/931
90	Trigger Configuration Register (BCTU_TRGCFG_30)	32	R/W	0000_0000h	36.6.3/931
94	Trigger Configuration Register (BCTU_TRGCFG_31)	32	R/W	0000_0000h	36.6.3/931
98	Trigger Configuration Register (BCTU_TRGCFG_32)	32	R/W	0000_0000h	36.6.3/931
9C	Trigger Configuration Register (BCTU_TRGCFG_33)	32	R/W	0000_0000h	36.6.3/931
A0	Trigger Configuration Register (BCTU_TRGCFG_34)	32	R/W	0000_0000h	36.6.3/931
A4	Trigger Configuration Register (BCTU_TRGCFG_35)	32	R/W	0000_0000h	36.6.3/931
A8	Trigger Configuration Register (BCTU_TRGCFG_36)	32	R/W	0000_0000h	36.6.3/931
AC	Trigger Configuration Register (BCTU_TRGCFG_37)	32	R/W	0000_0000h	36.6.3/931
B0	Trigger Configuration Register (BCTU_TRGCFG_38)	32	R/W	0000_0000h	36.6.3/931
B4	Trigger Configuration Register (BCTU_TRGCFG_39)	32	R/W	0000_0000h	36.6.3/931
B8	Trigger Configuration Register (BCTU_TRGCFG_40)	32	R/W	0000_0000h	36.6.3/931
BC	Trigger Configuration Register (BCTU_TRGCFG_41)	32	R/W	0000_0000h	36.6.3/931
C0	Trigger Configuration Register (BCTU_TRGCFG_42)	32	R/W	0000_0000h	36.6.3/931
C4	Trigger Configuration Register (BCTU_TRGCFG_43)	32	R/W	0000_0000h	36.6.3/931
C8	Trigger Configuration Register (BCTU_TRGCFG_44)	32	R/W	0000_0000h	36.6.3/931
CC	Trigger Configuration Register (BCTU_TRGCFG_45)	32	R/W	0000_0000h	36.6.3/931
D0	Trigger Configuration Register (BCTU_TRGCFG_46)	32	R/W	0000_0000h	36.6.3/931
D4	Trigger Configuration Register (BCTU_TRGCFG_47)	32	R/W	0000_0000h	36.6.3/931
D8	Trigger Configuration Register (BCTU_TRGCFG_48)	32	R/W	0000_0000h	36.6.3/931
DC	Trigger Configuration Register (BCTU_TRGCFG_49)	32	R/W	0000_0000h	36.6.3/931

Table continues on the next page...

BCTU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E0	Trigger Configuration Register (BCTU_TRGCFG_50)	32	R/W	0000_0000h	36.6.3/931
E4	Trigger Configuration Register (BCTU_TRGCFG_51)	32	R/W	0000_0000h	36.6.3/931
E8	Trigger Configuration Register (BCTU_TRGCFG_52)	32	R/W	0000_0000h	36.6.3/931
EC	Trigger Configuration Register (BCTU_TRGCFG_53)	32	R/W	0000_0000h	36.6.3/931
F0	Trigger Configuration Register (BCTU_TRGCFG_54)	32	R/W	0000_0000h	36.6.3/931
F4	Trigger Configuration Register (BCTU_TRGCFG_55)	32	R/W	0000_0000h	36.6.3/931
F8	Trigger Configuration Register (BCTU_TRGCFG_56)	32	R/W	0000_0000h	36.6.3/931
FC	Trigger Configuration Register (BCTU_TRGCFG_57)	32	R/W	0000_0000h	36.6.3/931
100	Trigger Configuration Register (BCTU_TRGCFG_58)	32	R/W	0000_0000h	36.6.3/931
104	Trigger Configuration Register (BCTU_TRGCFG_59)	32	R/W	0000_0000h	36.6.3/931
108	Trigger Configuration Register (BCTU_TRGCFG_60)	32	R/W	0000_0000h	36.6.3/931
10C	Trigger Configuration Register (BCTU_TRGCFG_61)	32	R/W	0000_0000h	36.6.3/931
110	Trigger Configuration Register (BCTU_TRGCFG_62)	32	R/W	0000_0000h	36.6.3/931
114	Trigger Configuration Register (BCTU_TRGCFG_63)	32	R/W	0000_0000h	36.6.3/931
228	Write Protection register (BCTU_WRPROT)	32	R/W	0000_0000h	36.6.4/933
22C	Software Trigger register 1 (BCTU_SFTRGR1)	32	R/W	0000_0000h	36.6.5/934
230	Software Trigger register 2 (BCTU_SFTRGR2)	32	R/W	0000_0000h	36.6.6/937
23C	ADC0 Result Data register (BCTU_ADC0DR)	32	R	0000_0000h	36.6.7/940
240	ADC1 result data register (BCTU_ADC1DR)	32	R	0000_0000h	36.6.8/941
24C	LIST Size Status register (BCTU_LISTSTAR)	32	R/W	0000_0020h	36.6.9/942
250	LIST Channel Address Register (BCTU_LISTCHR_0)	32	R/W	0000_0000h	36.6.10/ 943
254	LIST Channel Address Register (BCTU_LISTCHR_1)	32	R/W	0000_0000h	36.6.10/ 943
258	LIST Channel Address Register (BCTU_LISTCHR_2)	32	R/W	0000_0000h	36.6.10/ 943
25C	LIST Channel Address Register (BCTU_LISTCHR_3)	32	R/W	0000_0000h	36.6.10/ 943
260	LIST Channel Address Register (BCTU_LISTCHR_4)	32	R/W	0000_0000h	36.6.10/ 943
264	LIST Channel Address Register (BCTU_LISTCHR_5)	32	R/W	0000_0000h	36.6.10/ 943
268	LIST Channel Address Register (BCTU_LISTCHR_6)	32	R/W	0000_0000h	36.6.10/ 943
26C	LIST Channel Address Register (BCTU_LISTCHR_7)	32	R/W	0000_0000h	36.6.10/ 943
270	LIST Channel Address Register (BCTU_LISTCHR_8)	32	R/W	0000_0000h	36.6.10/ 943
274	LIST Channel Address Register (BCTU_LISTCHR_9)	32	R/W	0000_0000h	36.6.10/ 943
278	LIST Channel Address Register (BCTU_LISTCHR_10)	32	R/W	0000_0000h	36.6.10/ 943

Table continues on the next page...

BCTU memory map (continued)

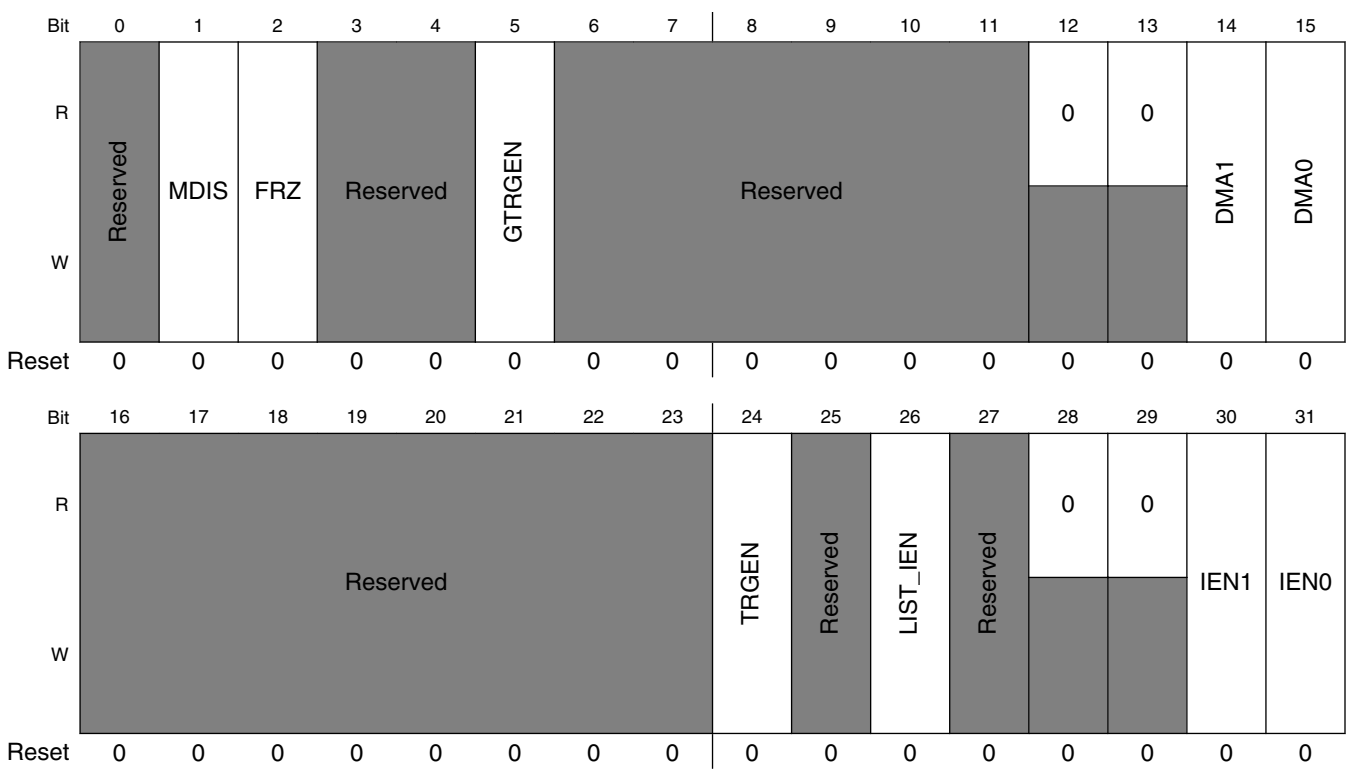
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
27C	LIST Channel Address Register (BCTU_LISTCHR_11)	32	R/W	0000_0000h	36.6.10/ 943
280	LIST Channel Address Register (BCTU_LISTCHR_12)	32	R/W	0000_0000h	36.6.10/ 943
284	LIST Channel Address Register (BCTU_LISTCHR_13)	32	R/W	0000_0000h	36.6.10/ 943
288	LIST Channel Address Register (BCTU_LISTCHR_14)	32	R/W	0000_0000h	36.6.10/ 943
28C	LIST Channel Address Register (BCTU_LISTCHR_15)	32	R/W	0000_0000h	36.6.10/ 943

36.6.1 Module Configuration Register (BCTU_MCR)

The Module Configuration register (MCR) provides access to low power, debug, trigger control, DMA control and Interrupt control bits.

Access: User read/write

Address: 0h base + 0h offset = 0h



BCTU_MCR field descriptions

Field	Description
0 Reserved	This field is reserved.
1 MDIS	Module Disable 0 Module is in normal mode. 1 Module is in low power mode.
2 FRZ	Freeze mode - for debug operation. The freeze mode disables all trigger inputs from timers but leave the software trigger enabled, thus allowing the software to trigger a conversion and read the conversion result.
3–4 Reserved	This field is reserved.
5 GTRGEN	Global Trigger Enable. Enables all trigger inputs at the same time. The individual trigger enable must be set in order for that trigger to become active.
6–11 Reserved	This field is reserved.
12 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 DMA1	ADC1DR data transferring by DMA. Enables DMA operation on transferring ADC1DR data when there is new data available in this register. 0 Disable DMA operation. 1 Enable DMA operation.
15 DMA0	ADC0DR data transferring by DMA. Enables DMA operation on transferring ADC0DR data when there is new data available in this register. 0 Disable DMA operation. 1 Enable DMA operation.
16–23 Reserved	This field is reserved.
24 TRGEN	Trigger interrupt request enable. Enables TRGF flag set in MSR register to generate interrupt request 0 Trigger interrupt request disabled. 1 Trigger interrupt request enabled. A request is generated if the TRGF flag bit in MSR register is set.
25 Reserved	This field is reserved.
26 LIST_IEN	LIST interrupt enable. If a LIST last conversion is executed an interrupt request is issued to the CPU if LIST_IEN is set. For the interrupt to occur the LIST and LAST flags must be set.

Table continues on the next page...

BCTU_MCR field descriptions (continued)

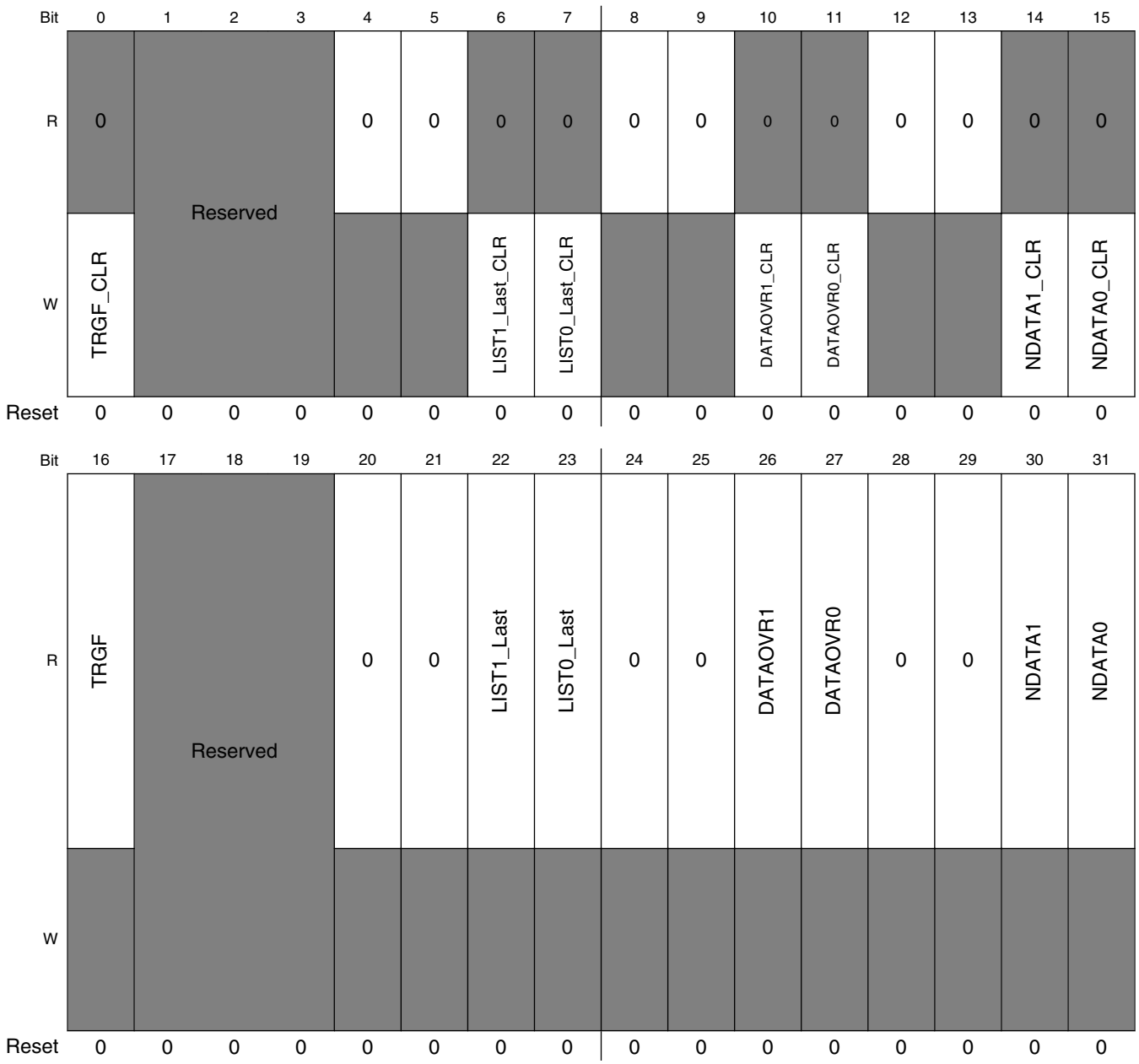
Field	Description
	0 No interrupt request. 1 Interrupt request.
27 Reserved	This field is reserved.
28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
30 IEN1	Interrupt enable bit for ADC1DR new data. This bit enables an interrupt request if the ADC1DR register is written by the hardware, meaning that there is new conversion data available. It also enables an interrupt request to be issued if there is an overrun in the ADC1DR register, meaning that the previous data in this register was overwritten by a new data. See also NDATA1 and DATAOVR1 in the MSR register. If DMA1 bit is set the interrupt for ADC1DR new data is disabled but the interrupt for ADC1DR data overrun remains enabled. In this case if an overrun condition occurs an interrupt request is generated. 0 Disable interrupt request. 1 Enable interrupt request.
31 IEN0	Interrupt enable bit for ADC0DR new data. This bit enables an interrupt request if the ADC0DR register is written by the hardware, meaning that there is new conversion data available. It also enables an interrupt request to be issued if there is an overrun in the ADC0DR register, meaning that the previous data in this register was overwritten by a new data. See also NDATA0 and DATAOVR0 in the MSR register. If DMA0 bit is set the interrupt for ADC0DR new data is disabled but the interrupt for ADC0DR data overrun remains enabled. In this case if an overrun condition occurs an interrupt request is generated. 0 Disable interrupt request. 1 Enable interrupt request.

36.6.2 Module Status register (BCTU_MSR)

The Status register provides software access to flags that indicate if conversion data is available for all ADCs. There are also indication of LIST status bits, such as last conversion command executed in a LIST.

Access: User read/write

Address: 0h base + 8h offset = 8h



BCTU_MSR field descriptions

Field	Description
0 TRGF_CLR	TRGF clear This bit is the clear for the TRGF flag bit 0 No action 1 Clears TRGF flag
1–3 Reserved	This field is reserved.

Table continues on the next page...

BCTU_MSR field descriptions (continued)

Field	Description
4 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
5 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
6 LIST1_Last_CLR	Clears LIST_Last flag for ADC1 0 no action 1 clears LIST1_Last flag
7 LIST0_Last_CLR	Clears LIST_Last flag for ADC0 0 no action 1 Clears LIST0_Last flag
8 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
9 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
10 DATAOVR1_ CLR	Clears DATAOVR1 flag 0 no action 1 Clears DATAOVR1 flag
11 DATAOVR0_ CLR	Clears DATAOVR0 flag 0 no action 1 Clears DATAOVR0 flag
12 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
14 NDATA1_CLR	Clears NDATA1 flag 0 no action 1 Clears NDATA1 flag
15 NDATA0_CLR	Clears NDATA0 flag 0 no action 1 Clears NDATA0 flag
16 TRGF	Trigger Flag

Table continues on the next page...

BCTU_MSR field descriptions (continued)

Field	Description
	Indicates that at least one ADC was triggered. 0 ADC was not triggered 1 An ADC was triggered
17–19 Reserved	This field is reserved.
20 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
22 LIST1_Last	Execution of the last conversion in a LIST of conversions for ADC1 0 not the last conversion 1 last conversion in a LIST
23 LIST0_Last	Execution of the last conversion in a LIST of conversions for ADC0 0 not the last conversion 1 last conversion in a LIST
24 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
25 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
26 DATAOVR1	Data overrun for ADC1. Old data in the ADCDR1 register was overwritten 0 data not overwritten 1 data overwritten
27 DATAOVR0	Data overrun for ADC0. Old data in the ADCDR0 register was overwritten 0 data not overwritten 1 data overwritten
28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
30 NDATA1	New conversion data is available for ADC1 0 no new data available 1 new data available

Table continues on the next page...

BCTU_MSR field descriptions (continued)

Field	Description
31 NDATA0	New conversion data is available for ADC0 0 no new data available 1 new data available

36.6.3 Trigger Configuration Register (BCTU_TRGCFG_*n*)

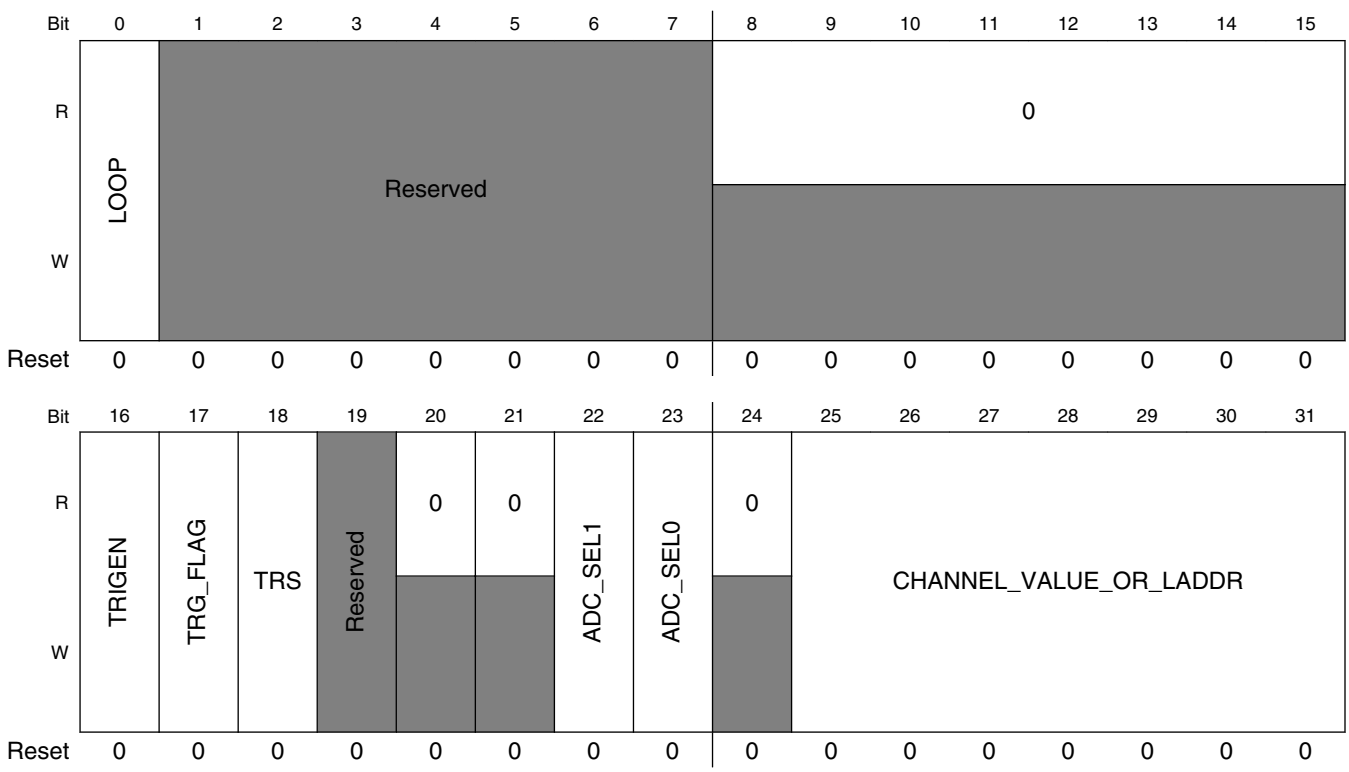
The Trigger Configuration register is a set of registers, one register per input trigger. This register controls the target ADC and ADC channel to be used in a conversion. It also controls if a LIST is used instead of a single ADC conversion command.

NOTE

x varies from 0 up to the number of input triggers.

Access: User read/write

Address: 0h base + 18h offset + (4d × i), where i=0d to 63d



BCTU_TRGCFG_*n* field descriptions

Field	Description
0 LOOP	LOOP bit

Table continues on the next page...

BCTU_TRGCFG_*n* field descriptions (continued)

Field	Description
	The LOOP bit , defined if the current trigger should execute a loop, meaning that the conversion triggered by the input trigger is executed again until the LOOP bit is cleared. Each time the new conversion is triggered it should win the priority evaluation among all pending triggers. It is only executed if it is the highest priority triggered conversion at the moment the conversion starts.
1–7 Reserved	This field is reserved.
8–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16 TRIGEN	Trigger enable Enable/disable triggers from the input trigger (hardware) to initiate an ADC conversion. This bit does not disable software triggers. 0 disable hardware triggers 1 enable hardware triggers
17 TRG_FLAG	Clear timer flag through software 0 When written, indicates no action. When read, indicates that there is no ongoing ADC conversion initiated by this trigger. 1 When written, clears timer flag through software. Which means that any previous input trigger event is cleared, preventing a new ADC conversion to be initiated by this trigger source. When read, indicates that there is an ongoing ADC conversion initiated by this trigger.
18 TRS	Trigger resolution Selects single conversion or LIST of conversions 0 single conversion 1 List of conversions
19 Reserved	This field is reserved.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 ADC_SEL1	Selects ADC 0 for conversion 0 does not selected ADC1 1 selects ADC1
23 ADC_SEL0	Selects ADC 0 for conversion 0 does not selected ADC0 1 selects ADC0
24 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

BCTU_TRGCFG_*n* field descriptions (continued)

Field	Description
25–31 CHANNEL_ VALUE_OR_ LADDR	Channel Value or List Address Value. If TRS = 0. this field is the ADC Channel selected for the conversion. If TRS = 1, this field represents the address of the first LIST position to be executed.

36.6.4 Write Protection register (BCTU_WRPROT)

Coming out from reset state the protection is enabled. In order to allow writes to protected registers it is necessary to write a special code in a 4-bit field. There are two possible codes that allow writes thus removing the protection. One 4-bit code that permanently disables the protection mechanism and another code that disables the protection during one write cycle to a protected register. After that the state of the protection code returns to its initial (protected) state.

Access: User read/write

Address: 0h base + 228h offset = 228h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved																												PROTEC_ CODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BCTU_WRPROT field descriptions

Field	Description
0–27 Reserved	This field is reserved.
28–31 PROTEC_CODE	The Protection code sets protection for the write protected registers There are two codes which disables the protection for the write protected registers. One code disables the write protection permanently and another code disables the protection for one write access only. **NOTE:** This field is writable only if the MCR[MDIS] bit is 0. 1010 This code defines that the protected registers are not in the protected mode, thus writes are enabled. 1001 Protection temporarily deactivated. This code disables the protection for writes on the protected registers for one write cycle operation. After that the protection code is reset to the initial state "0000" thus restoring the protection for the protected registers. Protection is active for other values

36.6.5 Software Trigger register 1 (BCTU_SFTRGR1)

The Software Trigger register allows the software to trigger any of the conversions associated with the Trigger Configuration registers. In order to avoid ambiguity in the conversion results regarding the source of trigger (if from hardware or from the software) the trigger enable (TRIGEN) bit in the Trigger Configuration register should be cleared when the software trigger is used thus the trigger for that configuration register is only issued by the SFTRGR register. This register is write protected by the WRPROT register. If protection is enabled writes to this register do not modify the register content and generates a transfer error.

All bits in this register are set by the software and cleared by the hardware.

Access: User read/write; this register is write protected by the BCTU_WRPROT register.

Address: 0h base + 22Ch offset = 22Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SFTRG31	SFTRG30	SFTRG29	SFTRG28	SFTRG27	SFTRG26	SFTRG25	SFTRG24	SFTRG23	SFTRG22	SFTRG21	SFTRG20	SFTRG19	SFTRG18	SFTRG17	SFTRG16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SFTRG15	SFTRG14	SFTRG13	SFTRG12	SFTRG11	SFTRG10	SFTRG9	SFTRG8	SFTRG7	SFTRG6	SFTRG5	SFTRG4	SFTRG3	SFTRG2	SFTRG1	SFTRG0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BCTU_SFTRGR1 field descriptions

Field	Description
0 SFTRG31	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
1 SFTRG30	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
2 SFTRG29	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

Table continues on the next page...

BCTU_SFTRGR1 field descriptions (continued)

Field	Description
3 SFTRG28	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
4 SFTRG27	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
5 SFTRG26	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
6 SFTRG25	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
7 SFTRG24	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
8 SFTRG23	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
9 SFTRG22	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
10 SFTRG21	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
11 SFTRG20	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
12 SFTRG19	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
13 SFTRG18	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
14 SFTRG17	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

Table continues on the next page...

BCTU_SFTRGR1 field descriptions (continued)

Field	Description
15 SFTRG16	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
16 SFTRG15	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
17 SFTRG14	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
18 SFTRG13	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
19 SFTRG12	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
20 SFTRG11	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
21 SFTRG10	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
22 SFTRG9	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
23 SFTRG8	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
24 SFTRG7	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
25 SFTRG6	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
26 SFTRG5	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

Table continues on the next page...

BCTU_SFTRGR1 field descriptions (continued)

Field	Description
27 SFTRG4	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
28 SFTRG3	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
29 SFTRG2	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
30 SFTRG1	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
31 SFTRG0	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

36.6.6 Software Trigger register 2 (BCTU_SFTRGR2)

The Software Trigger register allows the software to trigger any of the conversions associated with the Trigger Configuration registers. In order to avoid ambiguity in the conversion results regarding the source of trigger (if from hardware or from the software) the trigger enable (TRIGEN) bit in the Trigger Configuration register should be cleared when the software trigger is used thus the trigger for that configuration register is only issued by the SFTRGR register. This register is write protected by the WRPROT register. If protection is enabled writes to this register do not modify the register content and generates a transfer error.

All bits in this register are set by the software and cleared by the hardware.

Access: User read/write; this register is write protected by the BCTU_WRPROT register.

Address: 0h base + 230h offset = 230h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SFTRG63	SFTRG62	SFTRG61	SFTRG60	SFTRG59	SFTRG58	SFTRG57	SFTRG56	SFTRG55	SFTRG54	SFTRG53	SFTRG52	SFTRG51	SFTRG50	SFTRG49	SFTRG48
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SFTRG47	SFTRG46	SFTRG45	SFTRG44	SFTRG43	SFTRG42	SFTRG41	SFTRG40	SFTRG39	SFTRG38	SFTRG37	SFTRG36	SFTRG35	SFTRG34	SFTRG33	SFTRG32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

BCTU_SFTRGR2 field descriptions

Field	Description
0 SFTRG63	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
1 SFTRG62	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
2 SFTRG61	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
3 SFTRG60	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
4 SFTRG59	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
5 SFTRG58	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
6 SFTRG57	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
7 SFTRG56	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
8 SFTRG55	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
9 SFTRG54	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

Table continues on the next page...

BCTU_SFTRGR2 field descriptions (continued)

Field	Description
10 SFTRG53	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
11 SFTRG52	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
12 SFTRG51	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
13 SFTRG50	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
14 SFTRG49	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
15 SFTRG48	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
16 SFTRG47	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
17 SFTRG46	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
18 SFTRG45	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
19 SFTRG44	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
20 SFTRG43	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
21 SFTRG42	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

Table continues on the next page...

BCTU_SFTRGR2 field descriptions (continued)

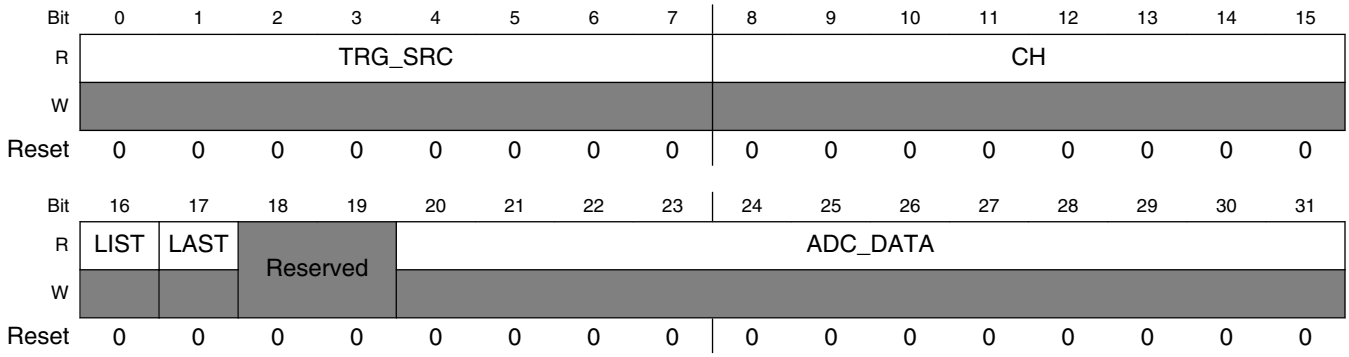
Field	Description
22 SFTRG41	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
23 SFTRG40	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
24 SFTRG39	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
25 SFTRG38	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
26 SFTRG37	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
27 SFTRG36	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
28 SFTRG35	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
29 SFTRG34	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger
30 SFTRG33	Software trigger to start an ADC conversion. 0 No effect 0 Set software trigger
31 SFTRG32	Software trigger to start an ADC conversion. 0 No effect 1 Set software trigger

36.6.7 ADC0 Result Data register (BCTU_ADC0DR)

The ADC Result Data register allows software access to ADC conversion result data. It also indicates the ADC channel used in that conversion, the trigger source that initiated that conversion and LIST status indication in case the conversion was initiated from a LIST.

Access: User read

Address: 0h base + 23Ch offset = 23Ch



BCTU_ADC0DR field descriptions

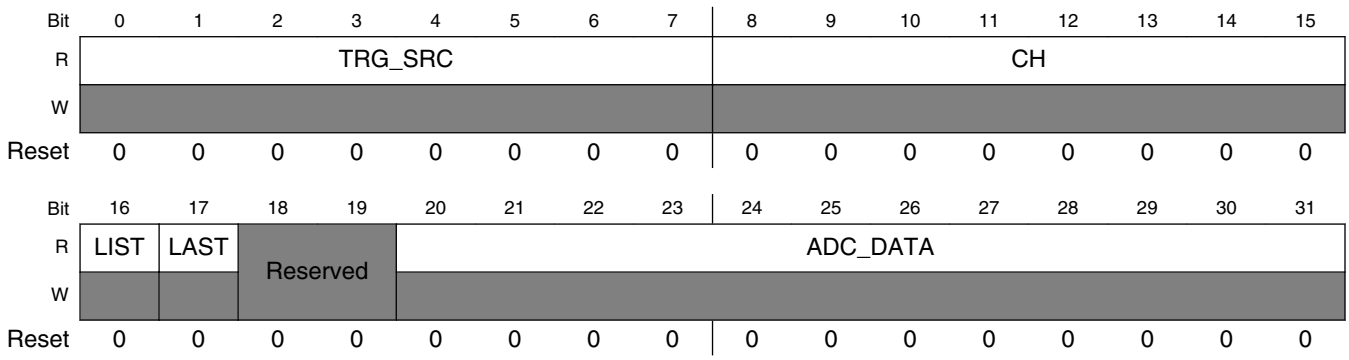
Field	Description
0–7 TRG_SRC	Trigger number used to triggered the conversion
8–15 CH	ADC Channel used for the conversion
16 LIST	Indicates if the conversion was initiated by a LIST element or single conversion trigger 0: single conversion 0 single conversion 1 LIST
17 LAST	In case the conversion was done through a LIST, if set to 1, this bit indicates that the conversion result is the last conversion in the LIST. If the conversion was not controlled by a LIST this bit is always zero. 0 The conversion result is not the last coversion in a LIST 1 The conversion result is the last conversion in a LIST
18–19 Reserved	This field is reserved.
20–31 ADC_DATA	DATA from ADC conversion

36.6.8 ADC1 result data register (BCTU_ADC1DR)

The ADC Result Data register allows software access to ADC conversion result data. It also indicates the ADC channel used in that conversion, the trigger source that initiated that conversion and LIST status indication in case the conversion was initiated from a LIST.

Access: User read

Address: 0h base + 240h offset = 240h



BCTU_ADC1DR field descriptions

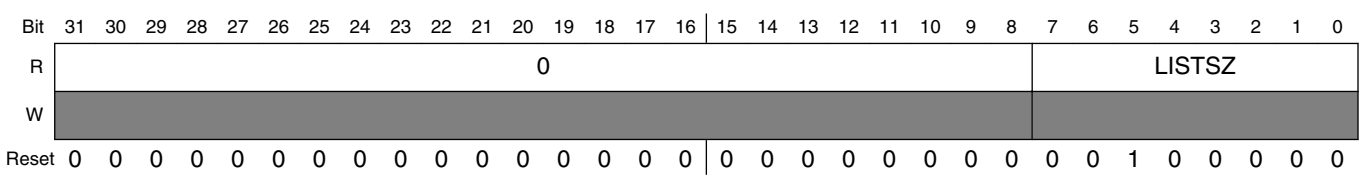
Field	Description
0–7 TRG_SRC	Trigger number used to triggered the conversion
8–15 CH	ADC Channel used for the conversion
16 LIST	Indicates if the conversion was initiated by a LIST element or single conversion trigger 0 single conversion 1 LIST
17 LAST	In case the conversion was done through a LIST, if set to 1, this bit indicates that the conversion result is the last conversion in the LIST. If the conversion was not controlled by a LIST this bit is always zero. 0 The conversion result is not the last coversion in a LIST 1 The conversion result is the last conversion in a LIST
18–19 Reserved	This field is reserved.
20–31 ADC_DATA	DATA from ADC conversion

36.6.9 LIST Size Status register (BCTU_LISTSTAR)

Each element has the channel number used in one conversion.

Access: User read

Address: 0h base + 24Ch offset = 24Ch



BCTU_LISTSTAR field descriptions

Field	Description
31–8 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
LISTSZ	Size of the LIST in number of elements. This field reflects the value of a hardware parameter.

36.6.10 LIST Channel Address Register (BCTU_LISTCHR_*n*)

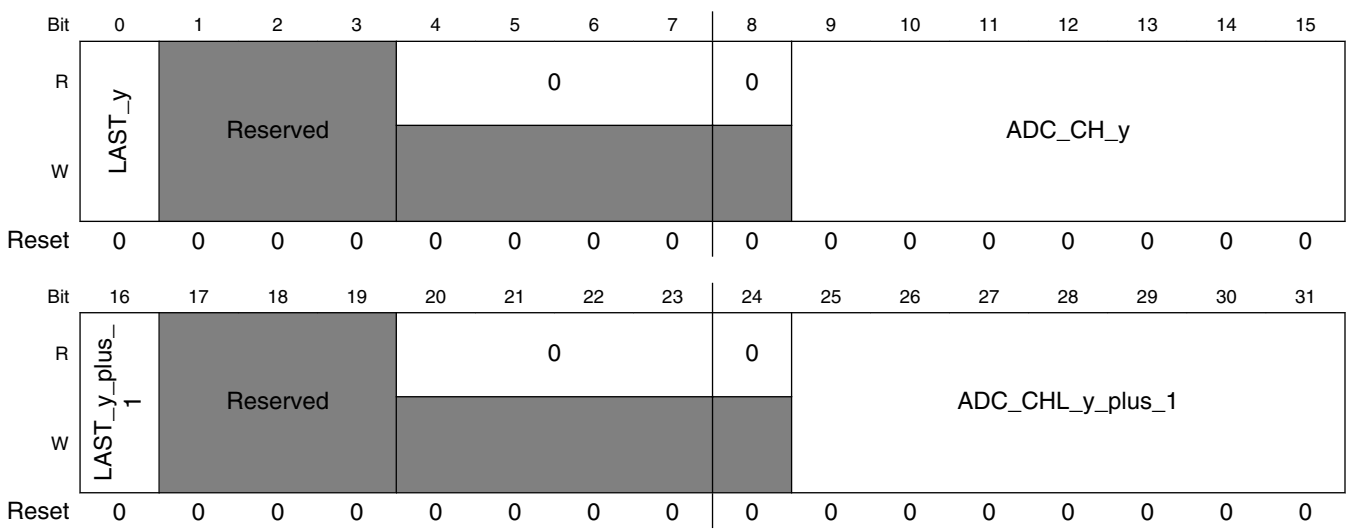
The LIST channel address register corresponds to two LIST elements which stores the ADC channel number to be used in a conversion as well as the indication if that element corresponds to the last element of a LIST of conversion commands.

NOTE

y varies from 0 to 31.

Access: User read/write

Address: 0h base + 250h offset + (4d × i), where i=0d to 15d



BCTU_LISTCHR_*n* field descriptions

Field	Description
0 LAST_y	Last channel in the list of channels. 0 It is not the last channel in the list 1 This is the last channel in the List
1–3 Reserved	This field is reserved.

Table continues on the next page...

BCTU_LISTCHR_*n* field descriptions (continued)

Field	Description
4–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
9–15 ADC_CH_y	ADC channel selection. y indicates the position (address) of the ADC channel in the LIST of channels.
16 LAST_y_plus_1	Second to last channel in the list of channels.
17–19 Reserved	This field is reserved.
20–23 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
24 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
25–31 ADC_CHL_y_ plus_1	ADC channel selection. y indicates the position (address) of the ADC channel in the LIST of channels.

36.7 Functional description

The following sections describes the BCTU functionality.

36.7.1 Trigger Configuration

The triggers from on-chip timers and software are configured in the TRGCFG (Trigger Configuration Register). Each trigger can be individually enabled by the TRIGEN bit in the TRGCFG_n registers. The target ADC and channel are also defined in this register. The TRS (Trigger Resolution bit) sets if either a single conversion is executed for the corresponding trigger or if the LIST of conversions should be executed. In the case of a LIST the channel/LADDR field is used as a pointer for the LIST element. If the LIST is used, multiple conversions are executed due to the corresponding trigger assertion.

A protocol is implemented such that when an input trigger is selected to issue an ADC conversion, the BCTU clears the corresponding trigger source when the ADC conversion is initiated.

NOTE

If a LIST is triggered the corresponding trigger will be cleared when the last conversion in the LIST is started.

The software trigger is controlled by the SFTRGR (Software Trigger Registers). By writing to a bit in those registers, an ADC conversion is triggered similar to a conversion initiated by an input trigger. A software triggered conversion does not require to be enabled by the TRIGEN bit in the TRGCFG register. Thus if any bit in the SFTRGR register is set, the corresponding conversion is triggered independent of the TRIGEN bit state. This functionality is useful such that it avoids conflict between software and hardware triggers.

NOTE

The software triggered conversion is enabled by the GTRGEN bit in the MCR register meaning that this bit needs to be set in order to allow a software conversion. This enable bit does not prevent the SFTRGR bits to be set, it just disables the conversion start. As soon as the GTRGEN bit is set the software conversions already triggered are initiated following the predefined priority scheme, i.e. from the lowest to the highest SFTRG bit number.

36.7.2 Trigger priority and multiplexing

The ADC triggers are prioritized based on the trigger number. The lower the trigger input number the higher is the priority associated with it. Thus TRG[0] is the highest priority trigger and TRG[95], in case 96 triggers are used, is the lowest priority. The trigger input priority scheme applies when the triggers occur at the same time or when the ADC is busy in a conversion. Thus there are two possible scenarios for the trigger priority selection: when there are no on-going conversions and when the ADC is executing a conversion.

The figure below describes the BCTU input triggers related to the ADC trigger signal generation. There is four clock cycles from the input trigger to the ADC trigger. This figure also shows the CLR[n] (clear) signal generated by the BCTU which clears the input trigger signal.

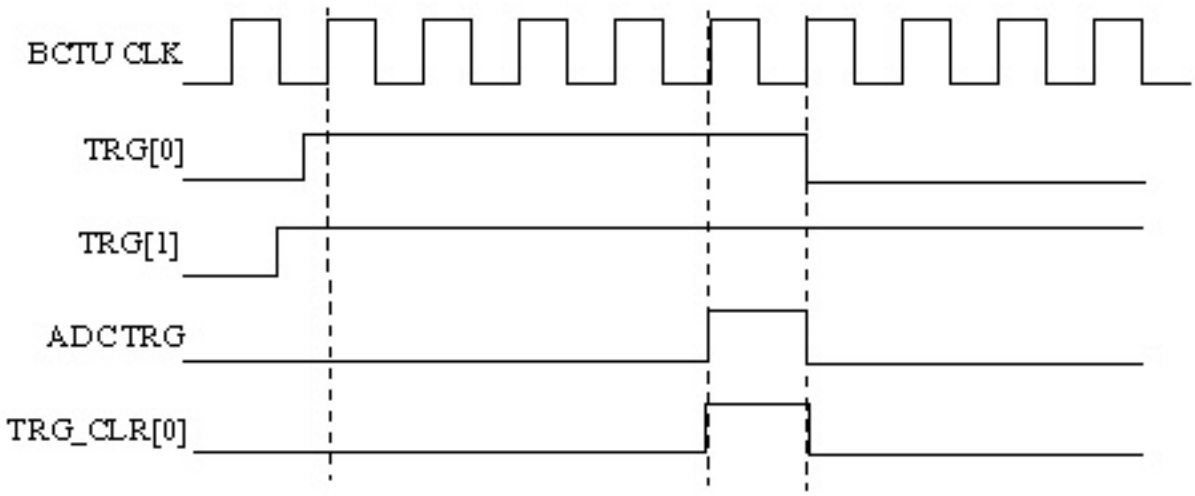


Figure 36-3. Input trigger to ADC trigger timing

The figure below describes the priority logic behavior in case TRG[0] and TRG[1] occur at the same time. Since TRG[0] has higher priority then it is selected for the ADC trigger generation. TRG[1] remains active (pending) until the conversion related to TRG[0] is executed. Note that if TRG[0] signal is asserted again, before the next trigger selection opportunity at the end of the current conversion, then it is selected one more time and TRG[1] remains asserted (pending) for more one ADC conversion time frame.

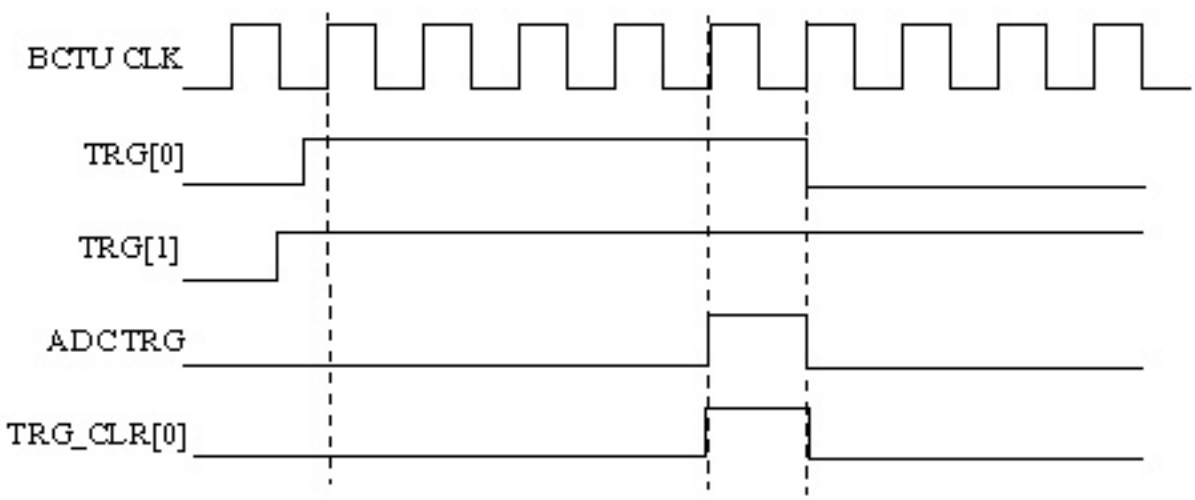


Figure 36-4. TRG[0] and TRG[1] priority example

The figure below describes the pipeline and logic (combinational logic) stages architecture, which resolves the priority among trigger inputs into two pipeline stages. The sel logic 1 and 2,combinational logic, are responsible for selecting the priority among all 96 inputs. The pipeline stage 3 stores this result and connects to a pulse generator stage which generates the trigger for the ADCs. Another stage isolates any

combinational logic in the BCTU to the external modules, thus allowing a clean clock domain crossing to connected to the ADC clock domain. In this architecture the timing from input trigger to the ADC trigger is 4 BCTU clock cycles.

NOTE

Back to back conversions are executed with 3 BCTU clock delay since the trigger selection is done during the current conversion thus the next trigger is ready when the conversion ends. The 4 clock delays occurs only when there are no conversions being executed by the ADC and new trigger occurs at the BCTU input.

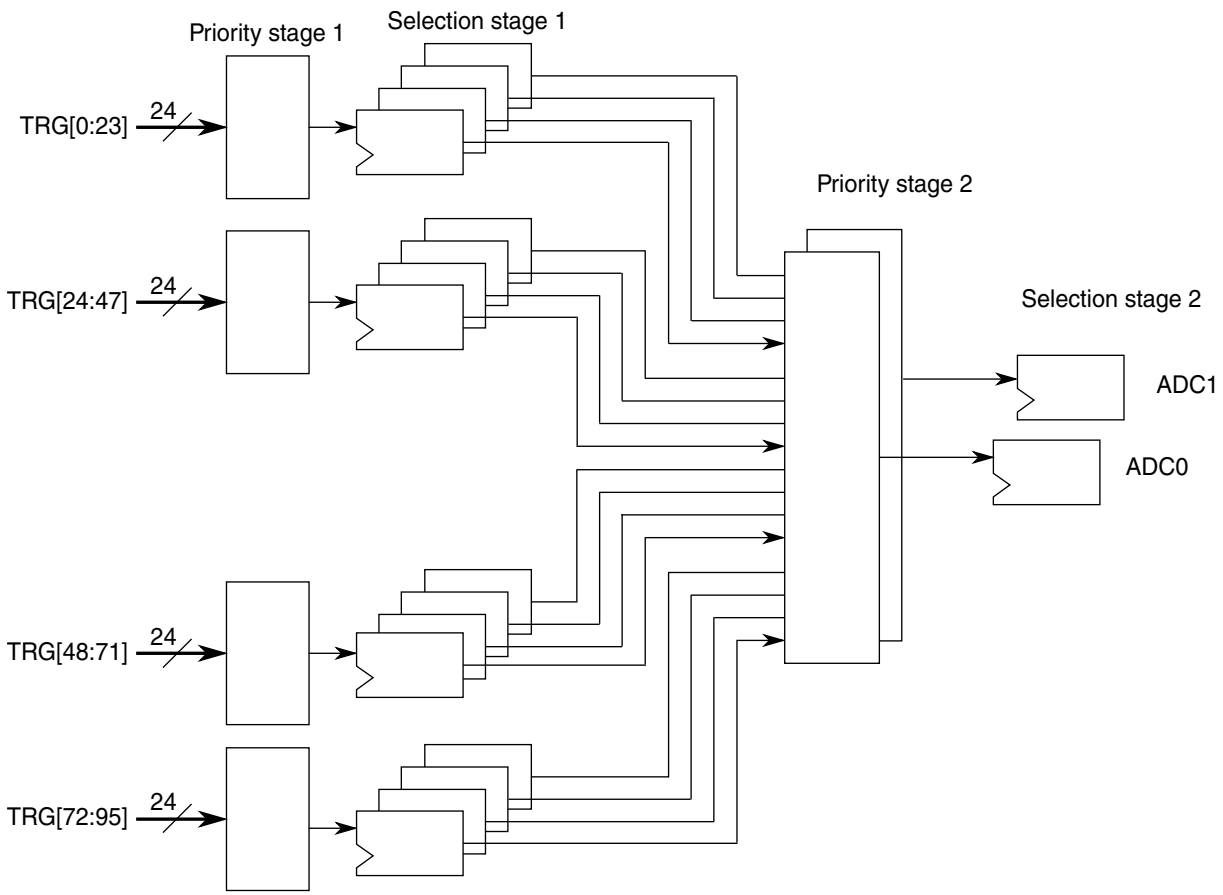
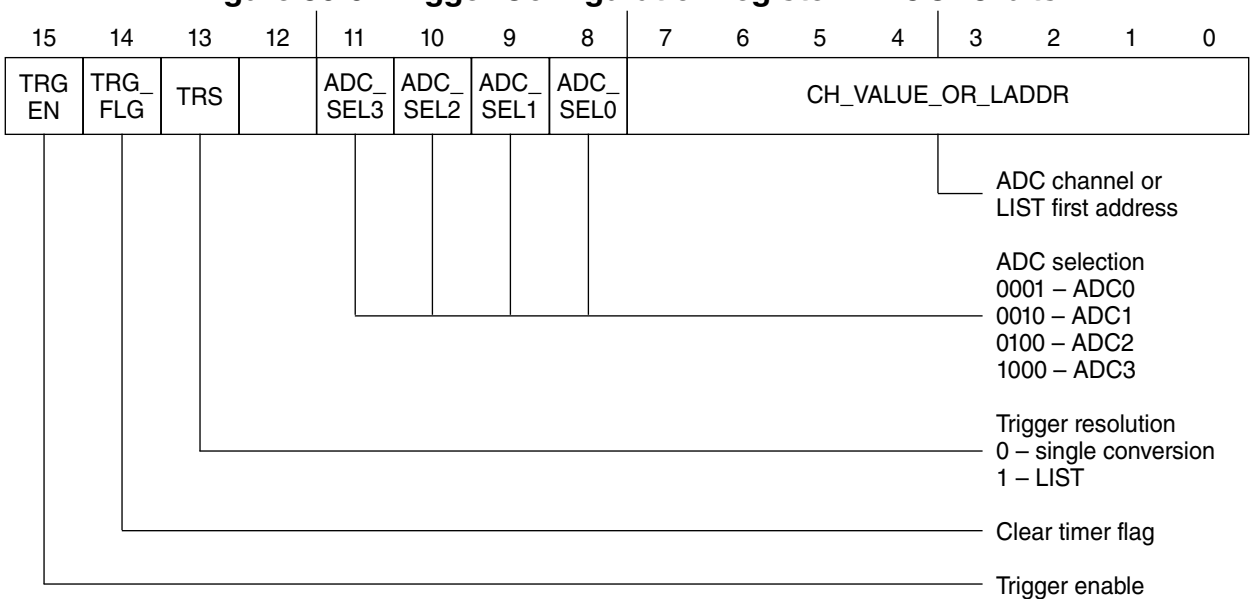


Figure 36-5. Priority stages architecture

36.7.3 Conversion LIST

The figure below describes the bits of the Trigger Configuration register (TRGCFG). The TRS bit selects between a single conversion and a list of conversions. In case a list of conversions is selected the CH_VALUE/LADDR field is used as the list pointer. The TRG_EN and CLR_FLAG bits have similar functionality regardless of single or list of conversion.

Figure 36-6. Trigger Configuration register -TRGCFG bits

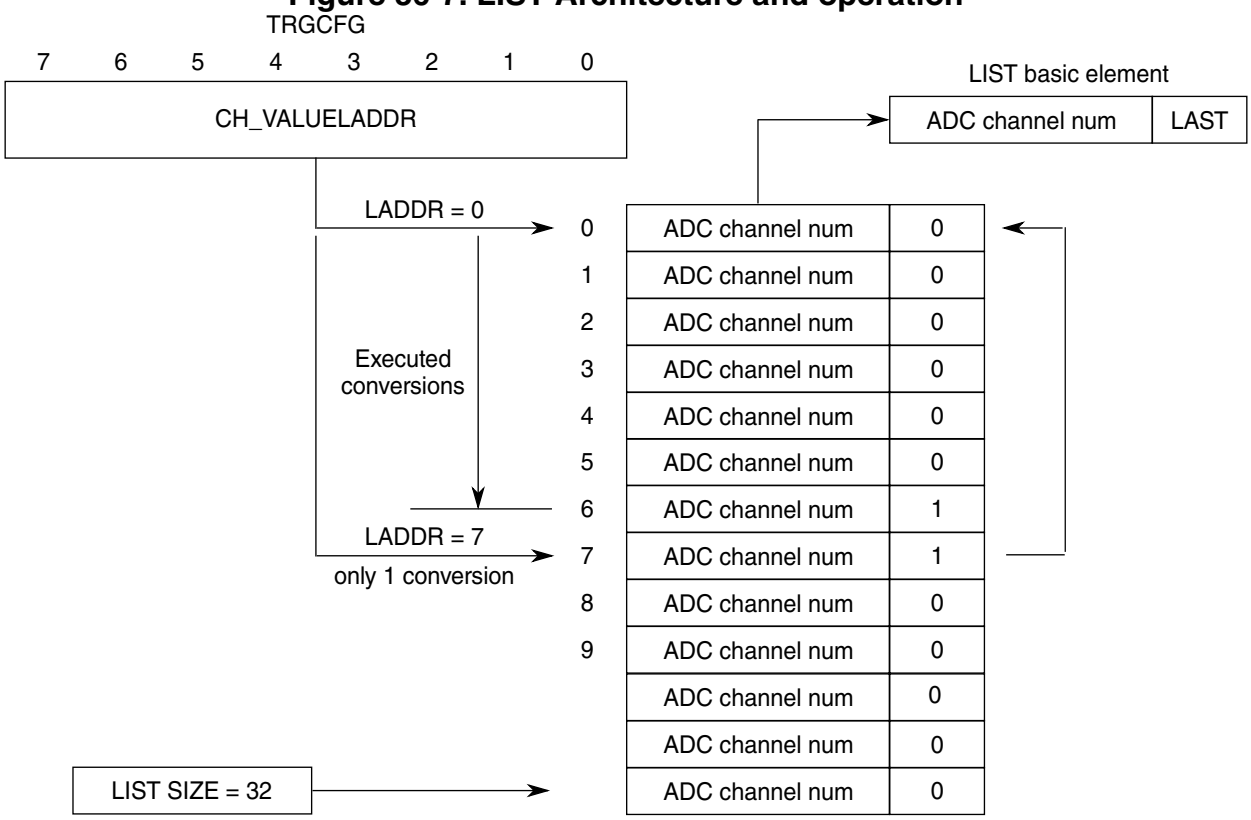


NOTE

ADC_SEL bits are one hot encoded, meaning that more than one ADC can be selected for a conversion. The multiple ADC selection is only valid for multiple parallel conversions LIST functionality. If more than one ADC is selected for a single conversion, instead of for a LIST, no conversion will be triggered by this register.

The figure below describes the LIST of channels architecture and operation. If the LADDR field in the TRGCFG register points to a LIST element then a sequence of conversions is initiated. The sequence of conversions are executed until the LAST field is set. The channel with the LAST field set is the last channel to be converted in a sequence of conversions.

Figure 36-7. LIST Architecture and operation



If LADDR points to a LIST element with the LAST field set, only one conversion is executed. In this case the LIST behaves such as a single conversion.

The LIST of channels has the following rules:

- A sequence of conversions is executed until the LAST field is evaluated as 1
- Conversions are executed in a sequence, thus other conversions for the same ADC are not allowed in parallel (the LIST is atomic)
- Two or more triggers for different ADCs may point to the LIST. In this case both LISTs are executed in parallel for the selected ADCs
- LADDR should point to a valid list address < LIST SIZE, otherwise address 0 is used
- After LIST SIZE -1 address is executed the list pointer wraps to address 0

36.7.4 ADC conversion results access

The access to ADC results can be done by the software using the flags and interrupt request or by the DMA interface. The interrupt request is based on the Status Register (MSR) flags. The results are available in the ADCxDR registers, one for each ADC. This register also indicates the channel used in the conversion and the trigger input that initiated that conversion. In order to give support for conversions from a LIST, two bits are available in the ADCxDR register, the LIST and LAST bits. The LIST bit indicates that the conversion was generated by a LIST element. The LAST bit indicates that the data is the last conversion of a LIST.

The software access to converted data can be based on interrupt request. An interrupt request is generated if the IENx bit in the MCR register is set and the corresponding DMAx bit is not set. In this case on each available conversion data in the ADCxDR register an interrupt request is asserted. The software can then check which is the source of the interrupt by reading the MSR register. Note that more then one conversion data may be available at the same time, since the conversions on different ADCs are executed in parallel. The NDATAx bit in the MSR register indicates which ADCxDR has a new conversion data available. The NDATAx flag can be cleared by using the NDATAx_CLR bit in the MSR register.

If DMA operation is selected by the DMA bit in the MCR register, a new data in the ADCxDR registers generates a DMA request. The NDATAx flag is also asserted in this case. A DMA DONE signal from the DMA controller de-asserts the DMA request and clears the NDATAx bit.

In order to allow DMA operation for all ADCxDR registers it is required one DMA channel per ADC, thus two DMA channels if two ADCs are used.

In case a new ADC conversion data is available before the old data had been read from the ADCxDR register then the DATAOVRx flag, new data overrun flag, is set indicating that the previous data in the ADCxDR register was overwritten by a new conversion result. The DATAOVRx flag also sets an interrupt request which is also enabled by the IENx bit. This interrupt request is asserted independently of the DMA bit settings. The DATAOVRx flag can only be cleared by the software writing 1 to the DATAOVRx_CLR bit.

36.7.5 Trigger clearing mechanism

There are two sources of ADC conversion triggers: hardware trigger, from the trigger inputs and software triggers, from SFTRGR register

A logical level 1 is detected in the input and captured into an input trigger register. This register remains set until the related conversion starts. At this time, the input trigger register is cleared. If another logical level 1 is detected while the trigger register is already set, the new level is not considered by the BCTU hardware. If a new logical level 1 occurs at the same time the trigger register is being cleared, then the register remains set and a new conversion is triggered by this input.

NOTE

The new triggered conversion will participate in the trigger priority selection thus may not trigger the next conversion.

The software trigger behaves similarly to the hardware trigger. The difference is that the trigger register is set by a CPU write operation. The trigger register is cleared by the BCTU hardware when the related conversion starts. If the CPU tries to set the trigger register by writing to the SFTRGR register corresponding bit, while the register is already set, the new attempt to set the register is not considered by the BCTU hardware. If a set occurs at the same time the hardware is attempting to clear the same trigger register, then the set has precedence and the register remains set. A new conversion will be triggered following the trigger priority criteria implemented in the hardware.

In case a LIST of conversions is triggered, instead of a single conversion, the trigger register is cleared at the start of the last conversion in the LIST.

36.7.6 On-the-fly reconfiguration of BCTU registers

The support for on-the-fly reconfiguration of triggers, specially for the Trigger Configuration register is done by the TRG_FLAG. If this bit field is read as 1, then the related trigger had started a conversion which is still ongoing. Therefore it is not safe to change the register bits.

The process of changing the Trigger Configuration register should start by clearing the TRIGEN bit. This will disable any new conversion be started by the corresponding trigger. After that the value of the TRG_FLAG bit should be evaluated. If this bit is set then it indicates that the current ongoing conversion or LIST execution was started by the trigger controlled by this register. The software should then read the TRGCFG register until the TRG_FLAG bit is cleared, which indicates that the on-going conversion had finished. After that the register can be modified safely. No conversions occur while the TRIGEN bit is cleared. In order to make sure that any trigger that occurred before the TRGCFG reconfiguration starts a new conversion, the application software should write 1 to the TRG_FLAG bit, thus clearing the trigger sources. Only triggers that occurred after this write operation was done are able to start a new conversion.

36.7.7 Multiple Parallel Conversions

The Multiple Parallel Conversions functionality is implemented using the LIST of channels. In order to execute this function more than one ADC should be selected in the TRGCFG register and the TRS bit must be set. In this case the TRGCFG CH_VALUE/ LADDR field points to an element in the LIST. The channels in the LIST are interpreted as referring to the ADCs selected in the TRGCFG register. The figure below describes the parallel conversion execution. Note that the channel numbers in the LIST are interpreted in pairs since two ADCs are selected, ADC0 and ADC1. If three ADC are selected, the LIST is interpreted in groups of three elements, being the order from ADC0 through ADC2.

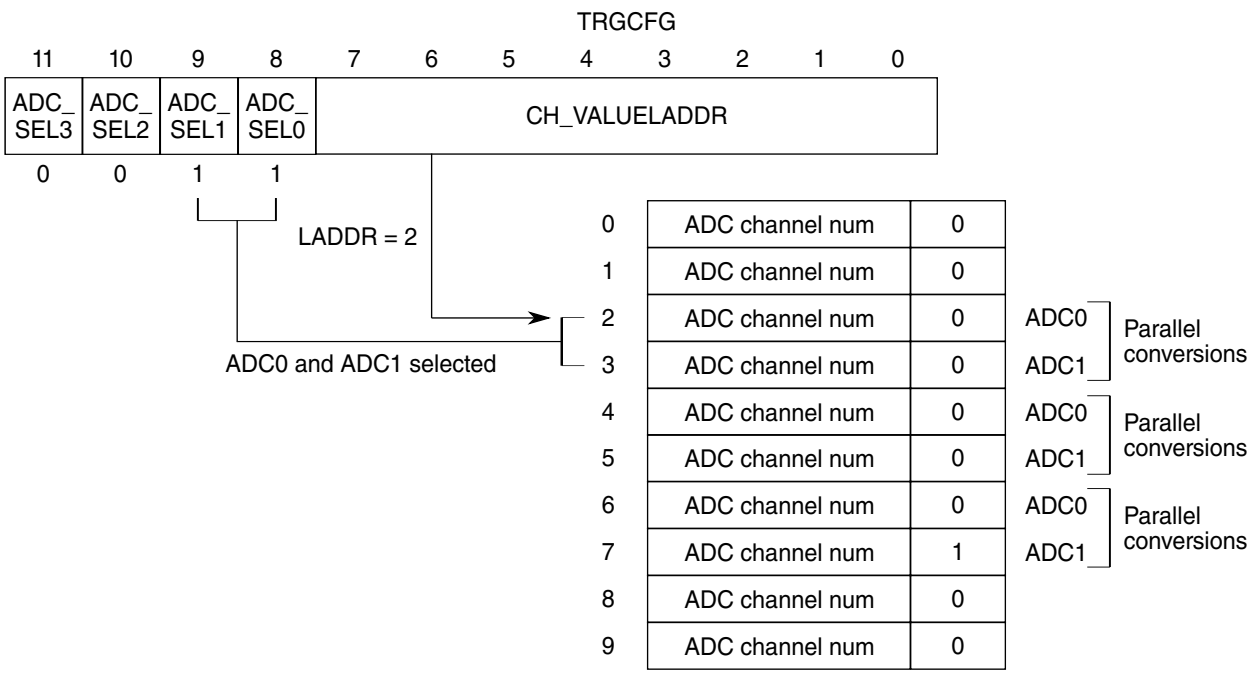


Figure 36-8. Multiple Parallel Conversions LIST operation

The Multiple Parallel Conversion (MPC) follows the same priority rules of a single conversion which is based on the trigger input number: the lower the trigger the higher the priority is. The difference is that a MPC cannot start until all the ADCs defined in the LIST are free. Thus if a MPC is triggered for ADC0 and ADC1, this LIST of conversions does not initiate until ADC0 and ADC1 are free. Note that if ADC0 is free but there is a higher priority trigger for ADC1 then the MPC does not start. If two MPC that have at least one ADC in common are triggered, the lower trigger number has higher priority. Note that the conflict in the priority only occurs if the triggered conversions are trying to use at least one ADC in common. If the MPCs are using different ADCs than there is no conflict and the MPCs are executed at the same time.

36.8 Interrupts and DMA requests

The BCTU module has several interrupts to signal input triggers and conversion results events. These interrupts can also be used as DMA requests.

Trigger interrupt : signals an ADC trigger event occurred, thus a conversion have started. This interrupt line is an or of all ADC triggers (up to two).

ADC data ready: these interrupts (up to two) signals when an ADC result is available. The software should read the BCTU_ADCx_DR register in order to access the ADC conversion result.

LIST interrupt: the LIST interrupt line signals when the LIST executed the last conversion. If the conversion from a LIST was transferred to the system memory then this interrupt signals to the software that all results, from a LIST, are ready to be processed. Note that this interrupt always occur along with the ADC data ready interrupt.

Chapter 37
Configurable Enhanced Modular IO Subsystem (eMIOS)

37.1 Chip-specific eMIOS information

37.1.1 eMIOS configuration

The device has 2 eMIOS modules that are each configured as shown in the following table.

Table 37-1. Configurations

Parameter	eMIOS_0	eMIOS_1
Timer width	16 bits	16 bits
Number of channels	32	32
Channels configuration	See EMIOS channel configuration figure below	
Channel pre-scaler width	4	4
Number of global counter buses	2	2
Number of global pre-scalers	1	1

37.1.2 Channel types

The eMIOS contain 4 different types of channel, which are listed below.

Table 37-2. eMIOS channel types

Mode Description	Name	Ch TypeX	Ch TypeY	Ch TypeG	Ch TypeH
General Purpose Input / Output	GPIO	X	X	X	X
Single Action Input Capture	SAIC	X	X	X	X

Table continues on the next page...

Table 37-2. eMIOS channel types (continued)

Mode Description	Name	Ch TypeX	Ch TypeY	Ch TypeG	Ch TypeH
Single Action Output Compare	SAOC	X	X	X	X
Modulus Counter	MC	X			
Modulus Counter Buffered (Up / Down)	MCB	X		X	
Input Pulse Width Measurement	IPWM			X	X
Input Period Measurement	IPM			X	X
Double Action Output Compare	DAOC			X	X
Output Pulse Width and Frequency Modulation Buffered	OPWFMB	X		X	
Center aligned Output PWM Buffered with dead time	OPWMCB			X	
Output Pulse Width Modulation Buffered	OPWMB	X	X	X	X
Output Pulse Width Modulation Trigger	OPWMT	X	X	X	X

NOTE

ChType X and G has internal counters.

Table 37-3. eMIOS channel configuration

Channel number	eMIOS0	eMIOS1
CH0	X	X
CH8	X	X
CH16	X	X
CH22	X	X
CH23	X	X
CH24	X	X
CH1	G	H
CH2	G	H
CH3	G	H
CH4	G	H
CH5	G	H
CH6	G	H

Table continues on the next page...

Table 37-3. eMIOS channel configuration (continued)

CH7	G	H
CH9	H	H
CH10	H	H
CH11	H	H
CH12	H	H
CH13	H	H
CH14	H	H
CH15	H	H
CH17	Y	Y
CH18	Y	Y
CH19	Y	Y
CH20	Y	Y
CH21	Y	Y
CH25	Y	Y
CH26	Y	Y
CH27	Y	Y
CH28	Y	Y
CH29	Y	Y
CH30	Y	Y
CH31	Y	Y

37.1.3 DMA channel configuration

Table 37-4. eMIOS DMA Channel configuration

Channel type	Channel number
eMIOSx_chA	Channel 0
eMIOSx_chB	Channel 1
eMIOSx_chC	Channel 9
eMIOSx_chD	Channel 10
eMIOSx_chE	Channel 17
eMIOSx_chF	Channel 18
eMIOSx_chG	Channel 25
eMIOSx_chH	Channel 26

37.1.4 Channel configuration

The eMIOS channel configuration is shown below. Please note that each channel and eMIOS are not identical.

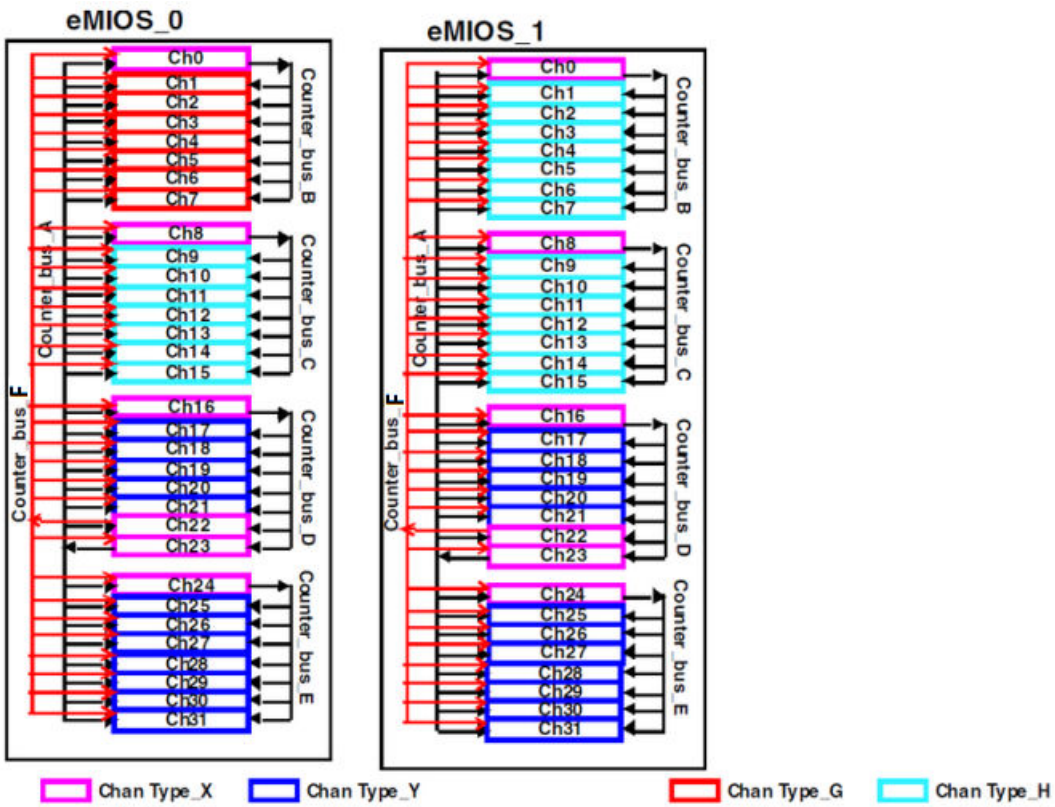


Figure 37-1. EMIOS channel types

37.1.5 eMIOS disabling

eMIOS can disable it's output using it's disable inputs. Disable inputs[3:0] are driven from the flag out bits[11:8].

Disable inputs	
eMIOS0 Output disable input[3:0]	eMIOS0 flag_out bits[11:8]
eMIOS1 Output disable input[3:0]	eMIOS1 flag_out bits[11:8]

37.1.6 Serialization

The channels from eMIOS_1 and eMIOS_0 are serialized with the DSPI[0:3] to allow control of external Lamp/LED drivers. PISR muxes are integrated into the DSPIs and also DSPI[0:1] supports DSI 64 bit mode. Hence routing between eMIOS and DSPIs are different. See diagram below for details

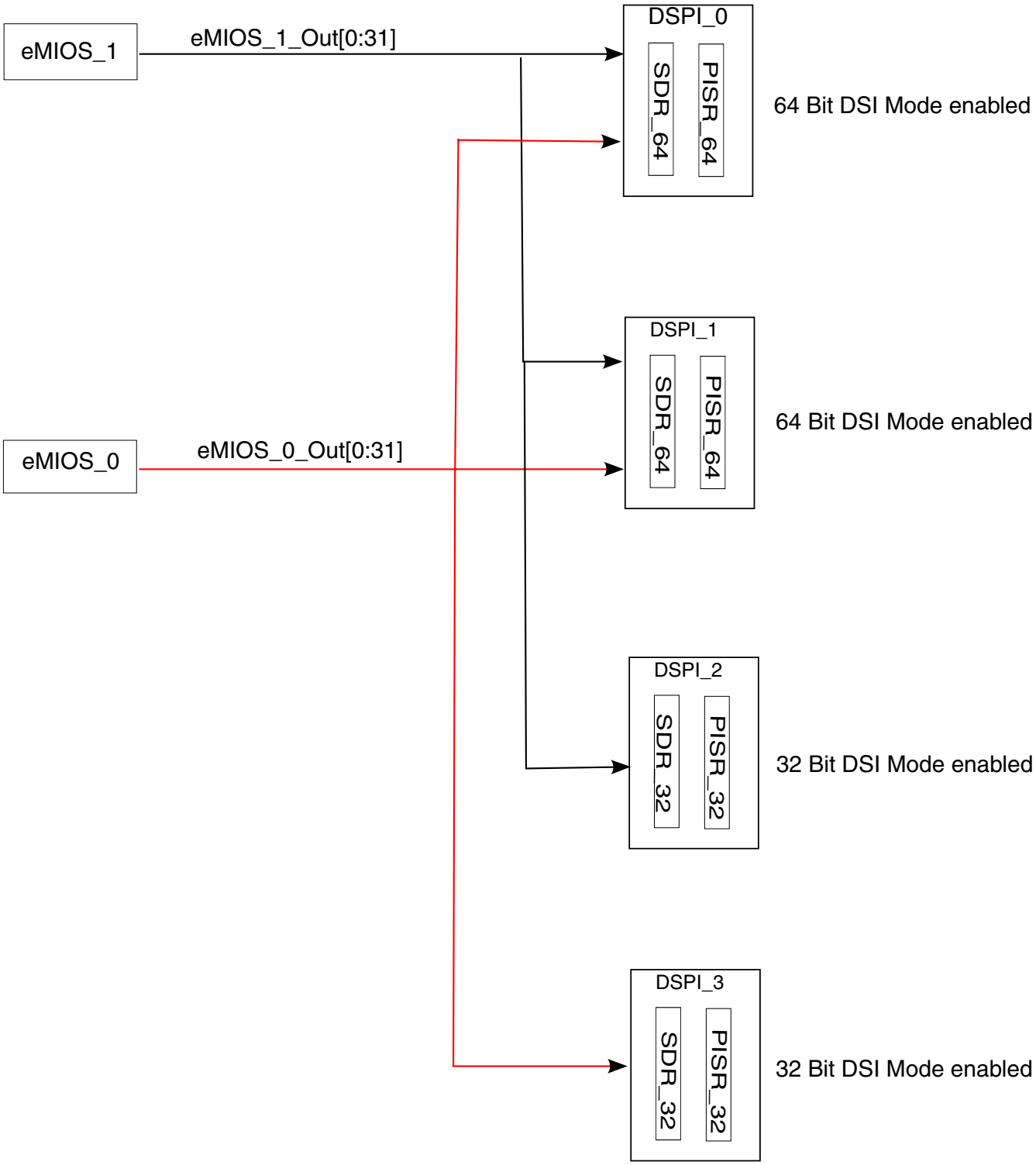


Figure 37-2. eMIOS serialisation

Table 37-5. eMIOS channel selection - 32 bit mode

PISR number (DSPIx_PISRn)			IPSx Mux field	IPSx field value and integer equivalent															
				8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7
				-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
eMIOS1 to DSPI0	eMIOS1 to DSPI1	eMIOS1 to DSPI2		Resultant eMIOS channel number															
0	0	0	0	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7
0	0	0	1	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8
0	0	0	2	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9
0	0	0	3	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10
0	0	0	4	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	5	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	6	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	7	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	9	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	10	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	1	11	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	1	12	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	1	1	13	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1	14	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	1	1	15	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	2	2	16	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
2	2	2	17	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
2	2	2	18	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	2	2	19	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	2	2	20	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
2	2	2	21	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
2	2	2	22	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
2	2	2	23	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Table continues on the next page...

Table 37-5. eMIOS channel selection - 32 bit mode (continued)

3	3	3	24	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
3	3	3	25	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0
3	3	3	26	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1
3	3	3	27	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2
3	3	3	28	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3
3	3	3	29	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4
3	3	3	30	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5
3	3	3	31	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6

Table 37-6. eMIOS channel selection - 64 bit mode

PISR Number	IPSx Mux Field	IPSx Field Value and Integer equivalent															
eMIOS2/eMIOS1 to DSPIx[1]		8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7
		-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
		Resultant EMIOS Channel Number															
0	0	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7
0	1	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8
0	2	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9
0	3	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10
0	4	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11
0	5	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12
0	6	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	7	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	9	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	10	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	11	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	12	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	13	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	14	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	15	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	16	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
2	17	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
2	18	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	19	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Table continues on the next page...

Table 37-6. eMIOS channel selection - 64 bit mode (continued)

2	20	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
2	21	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
2	22	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
2	23	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
3	24	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
3	25	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0
3	26	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1
3	27	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2
3	28	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3
3	29	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4
3	30	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5
3	31	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6
4	32	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7
4	33	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8
4	34	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9
4	35	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10
4	36	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11
4	37	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12
4	38	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13
4	39	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
5	40	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	41	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
5	42	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
5	43	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
5	44	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
5	45	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
5	46	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
5	47	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
6	48	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
6	49	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
6	50	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
6	51	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
6	52	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
6	53	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
6	54	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
6	55	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
7	56	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
7	57	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0
7	58	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1
7	59	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2

Table continues on the next page...

Table 37-6. eMIOS channel selection - 64 bit mode (continued)

7	60	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3
7	61	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4
7	62	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5
7	63	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6
Shaded Blocks Correspond to eMIOS1 Channels																	
Non-Shaded Blocks Correspond to eMIOS2 Channels																	

1. x is 0 or 1

37.1.7 BCTU Interface

Each eMIOS channel, with the exception of the main counterbus channel (ch[23]) is capable of triggering the BCTU. See BCTU chapter for mapping. BCTU allows triggering from as many eMIOS modes as possible.

37.1.8 EMIOS register reset value

All register reset values for the EMIOS registers that are device specific are shown below. All other registers reset values are shown in the EMIOS Memory chapter

EMIOS 0 disable channel register (EMIOS_0_UCDIS) resets to 0000_0000h.

EMIOS 1 disable channel register (EMIOS_1_UCDIS) resets to 0000_0000h.

37.2 Introduction

37.2.1 Overview

The configurable enhanced Modular Input/Output Subsystem (eMIOS) provides functionality to generate or measure time events. Its overall architecture resembles that of the MIOS timer.

The predecessor MIOS timer block provides a framework in which a set of sublocks with different timer functions are assembled to attend to the specific needs of an SoC. The eMIOS builds on this concept by using a Unified Channel module that provides a superset of the functionality of all the individual MIOS channels, while also providing a consistent user interface. This allows more flexibility as each Unified Channel can be

programmed for different functions in different applications of the SoC. In addition, eMIOS architecture allows the use of Dedicated Channels which perform specific functions not included in MIOS inheritance.

The block diagram of the configurable eMIOS block is shown below.

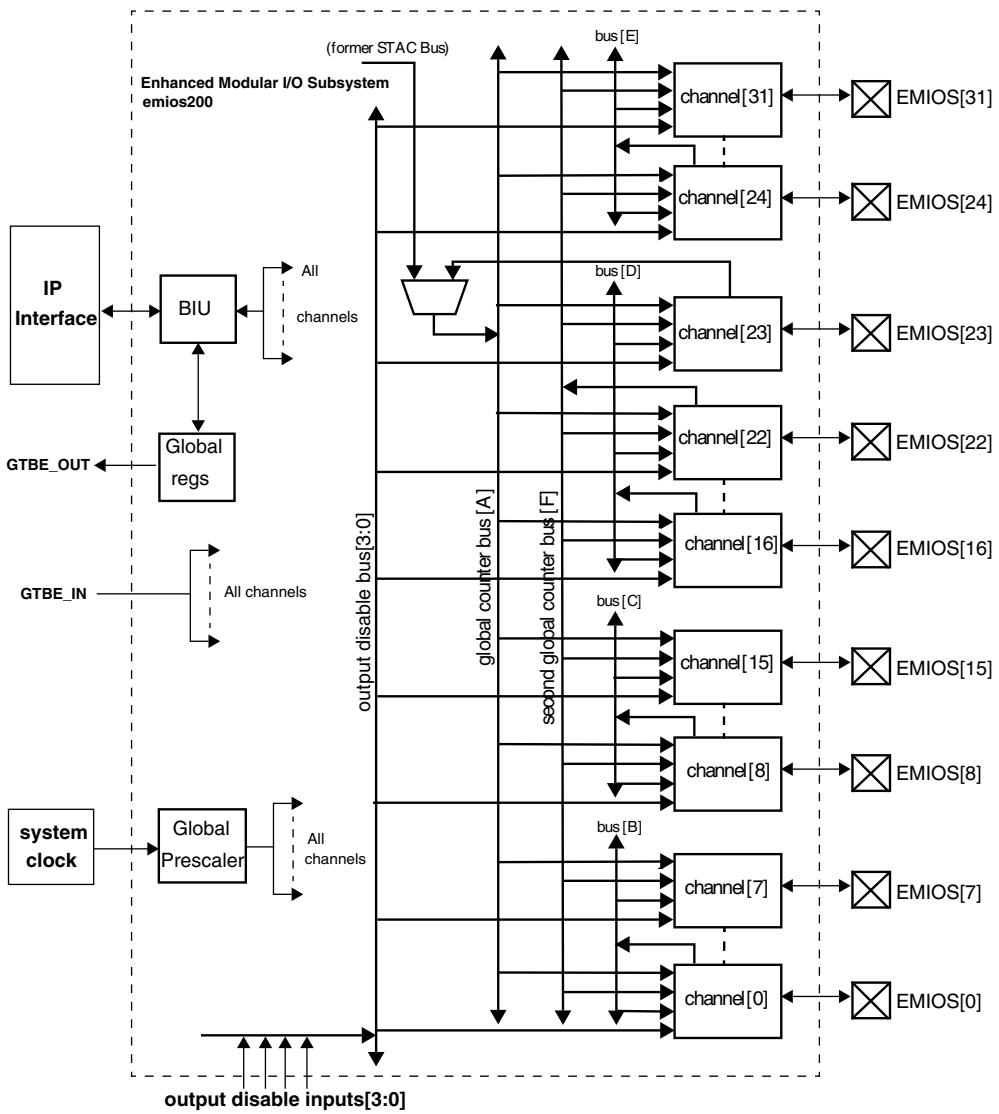


Figure 37-3. eMIOS block diagram with 2 global counter buses

37.2.2 Features

The eMIOS has the following features:

- Up to 32 channels chosen among Unified or Dedicated Channels, not necessarily numbered in a continuous sequence. (See Configuration information)
- Data registers of either 8-, 16-, 24-, or 32-bit width. (See Configuration information)
- Counter buses B, C, D, and E can be driven by Unified Channels 0, 8, 16, and 24, respectively. (See Configuration information)
- Counter bus A can be driven by Unified Channel 23 or by the external shared timer bus.
- Counter bus F can be driven by a specified Unified Channel, defined by the system configuration.
- Each channel has its own timebase, alternative to the counter buses.
- One global prescaler
- One prescaler per channel (CP)
- Timebases shared through the counter buses
- Control and status bits grouped in separated registers
- Synchronization among timebases
- Shadow FLAG register
- State of the Unified Channels can be frozen for debug purposes

37.2.3 Modes of operation

The Unified Channels can be configured to operate in the following modes:

- General-Purpose Input/Output (GPIO)
- Single-Action Input Capture (SAIC)
- Single-Action Output Compare (SAOC)
- Input Pulse-Width Measurement (IPWM)
- Input Period Measurement (IPM)

- Double-Action Output Compare (DAOC)
- Modulus Counter (MC)
- Modulus Counter Buffered (MCB)
- Output Pulse-Width and Frequency Modulation Buffered (OPWFMB)
- Center Aligned Output Pulse Width Modulation with Dead Time Insertion Buffered (OPWMCB)
- Output Pulse-Width Modulation Buffered (OPWMB)
- Output Pulse-Width Modulation with Trigger (OPWMT)

These modes are described in Unified Channel modes of operation.

Each channel can have a specific set of modes.

37.3 Memory map / register definition

37.3.1 Memory map

The overall address map organization is shown in the table below.

Whenever an access to either an absent register, absent channel, or reserved address space is performed, the transfer is terminated with an error.

Table 37-7. eMIOS Memory Map

eMIOS[n] Base Address	Description
0x000 0x003	Module Configuration register (EMIOSMCR)
0x004 0x007	Global FLAG register (EMIOSGFLAG)
0x008 0x00B	Output Update Disable (EMIOSOUDIS)
0x00C 0x00F	Disable Channel (EMIOSUCDIS)
0x01F	Reserved
0x020 0x11F	Channel 0 to Channel 7
0x120 0x21F	Channel 8 to Channel 15

Table continues on the next page...

Table 37-7. eMIOS Memory Map (continued)

eMIOS[n] Base Address	Description
0x220 0x31F	Channel 16 to Channel 23
0x320	Channel 24 to Channel 31
0x41F	
0x420	Reserved
0xFFF	

37.3.1.1 Unified channel memory map

Addresses of Unified Channel registers are specified as offsets from the channel's base address, otherwise the eMIOS base address is used as reference.

The following table describes the Unified Channel memory map.

Table 37-8. Unified channel memory map

UC[n] Base Address	Description
UC[n]_BASE + 0x000	A register (EMIOSA[n])
UC[n]_BASE + 0x004	B register (EMIOSB[n])
UC[n]_BASE + 0x008	Counter register (EMIOSCNT[n])
UC[n]_BASE + 0x00C	Control register (EMIOSC[n])
UC[n]_BASE + 0x010	Status register (EMIOSS[n])
UC[n]_BASE + 0x014	Alternate A register (EMIOSALTA[n])
UC[n]_BASE + 0x018	Control 2 register (EMIOSC2[n])
UC[n]_BASE + 0x01C to UC[n]_BASE + 0x01F	Reserved

37.4 Control Memory Map and Registers

All control registers are 32 bits wide. This document illustrates the eMIOS with 24 Unified Channels and 24-bit wide data registers.

eMIOS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	eMIOS Module Configuration Register (eMIOS_MCR)	32	R/W	0000_0000h	37.4.1/970

Table continues on the next page...

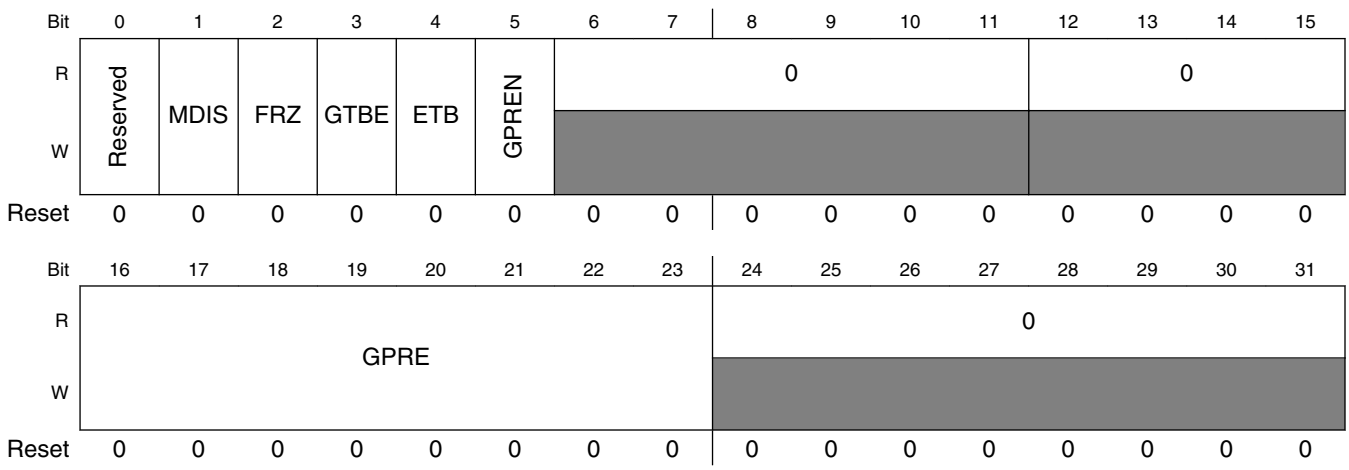
eMIOS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	eMIOS Global FLAG Register (eMIOS_GFLAG)	32	R	0000_0000h	37.4.2/972
8	eMIOS Output Update Disable register (eMIOS_OUDIS)	32	R/W	0000_0000h	37.4.3/972
C	eMIOS Disable Channel register (eMIOS_UCDIS)	32	R/W	0000_0000h	37.4.4/973

37.4.1 eMIOS Module Configuration Register (eMIOS_MCR)

The MCR register contains Global Control bits for the eMIOS block.

Address: 0h base + 0h offset = 0h



eMIOS_MCR field descriptions

Field	Description
0 Reserved	Reserved. This field is reserved. Only write this field with the reset value.
1 MDIS	Module Disable bit. Puts the eMIOS into low power mode. The MDIS bit stops the clock of the block, except the access to registers EMIOSMCR, EMIOSOUDIS and EMIOSUCDIS. 0 Clock is running 1 Enter low power mode
2 FRZ	Freeze bit. This bit enables the eMIOS to freeze the registers of the Unified Channels when Debug Mode is requested at MCU level. Each Unified Channel should have FREN bit set in order to enter freeze state. While in Freeze state, the eMIOS continues to operate to allow the MCU access to the Unified Channels registers. The Unified Channel will remain frozen until the FRZ bit is written to zero or the MCU exits Debug mode or the Unified Channel FREN bit is cleared.

Table continues on the next page...

eMIOS_MCR field descriptions (continued)

Field	Description
	0 Exit freeze state 1 Stops Unified Channels operation when in Debug mode and the FREN bit is set in the EMIOSC[n] register
3 GTBE	Global Time Base Enable bit. The GTBE bit is used to export a Global Time Base Enable from the module and provide a method to start time bases of several blocks simultaneously. **NOTE:** The Global Time Base Enable input pin controls the internal counters. When asserted, Internal counters are enabled. When negated, Internal counters disabled. 0 Global Time Base Enable Out signal negated 1 Global Time Base Enable Out signal asserted
4 ETB	External Time Base bit The ETB bit selects the time base source that drives counter bus[A]. 0 Counter bus[A] assigned to Unified Channel. 1 Reserved
5 GPREN	Global Prescaler Enable bit. The GPREN bit enables the prescaler counter. 0 Prescaler disabled (no clock) and prescaler counter is cleared 1 Prescaler enabled
6–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
12–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–23 GPRE	Global Prescaler bits. The GPRE[0:7] bits select the clock divider value for the global prescaler, as shown in the following table. GPRE values and corresponding divide ratio values are: 00000000 1 00000001 2 00000010 3 00000011 4 ------- - 11111110 255 11111111 256
24–31 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

37.4.2 eMIOS Global FLAG Register (eMIOS_GFLAG)

GFLAG is a read-only register that groups the FLAG bits from all channels. Each bit relates to one channel.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	F31_F0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_GFLAG field descriptions

Field	Description
0–31 F31_F0	Channel [n] mirrored FLAG bit. For Unified Channels these bits are mirrors of the FLAG bits in the EMIOSS[n] register.

37.4.3 eMIOS Output Update Disable register (eMIOS_OUDIS)

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	OU31_OU0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_OUDIS field descriptions

Field	Description
0–31 OU31_OU0	Channel [n] Output Update Disable bit. When running MC, MCB or an output mode, values are written to registers A2 and B2. OU[n] bits are used to disable transfers from registers A2 to A1 and B2 to B1. Each bit controls one channel.

37.4.4 eMIOS Disable Channel register (eMIOS_UCDIS)

NOTE

See the Configuration information for the reset value of this register.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CHDIS31_CHDIS0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_UCDIS field descriptions

Field	Description
0–31 CHDIS31_ CHDIS0	Enable Channel n bit. The CHDISn bit is used to disable each of the channels by stopping its respective clock.

37.5 Unified Channels Memory Map and Registers

The Unified Channel registers are as follows:

eMIOS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
20	eMIOS UC A register n (eMIOS_A0)	32	R/W	0000_0000h	37.5.1/979
24	eMIOS UC B register n (eMIOS_B0)	32	R/W	0000_0000h	37.5.2/980
28	eMIOS CNT register n (eMIOS_CNT0)	32	R/W	0000_0000h	37.5.3/981
2C	eMIOS UC Control register n (eMIOS_C0)	32	R/W	0000_0000h	37.5.4/982
30	eMIOS UC Status register n (eMIOS_S0)	32	R/W	See section	37.5.5/988
34	eMIOS Alternate Address register n (eMIOS_ALTA0)	32	R/W	0000_0000h	37.5.6/989
38	eMIOS UC Control 2 register n (eMIOS_C2_0)	32	R/W	0000_0000h	37.5.7/990
40	eMIOS UC A register n (eMIOS_A1)	32	R/W	0000_0000h	37.5.1/979
44	eMIOS UC B register n (eMIOS_B1)	32	R/W	0000_0000h	37.5.2/980
48	eMIOS CNT register n (eMIOS_CNT1)	32	R/W	0000_0000h	37.5.3/981
4C	eMIOS UC Control register n (eMIOS_C1)	32	R/W	0000_0000h	37.5.4/982
50	eMIOS UC Status register n (eMIOS_S1)	32	R/W	See section	37.5.5/988

Table continues on the next page...

eMIOS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
54	eMIOS Alternate Address register n (eMIOS_ALTA1)	32	R/W	0000_0000h	37.5.6/989
58	eMIOS UC Control 2 register n (eMIOS_C2_1)	32	R/W	0000_0000h	37.5.7/990
60	eMIOS UC A register n (eMIOS_A2)	32	R/W	0000_0000h	37.5.1/979
64	eMIOS UC B register n (eMIOS_B2)	32	R/W	0000_0000h	37.5.2/980
68	eMIOS CNT register n (eMIOS_CNT2)	32	R/W	0000_0000h	37.5.3/981
6C	eMIOS UC Control register n (eMIOS_C2)	32	R/W	0000_0000h	37.5.4/982
70	eMIOS UC Status register n (eMIOS_S2)	32	R/W	See section	37.5.5/988
74	eMIOS Alternate Address register n (eMIOS_ALTA2)	32	R/W	0000_0000h	37.5.6/989
78	eMIOS UC Control 2 register n (eMIOS_C2_2)	32	R/W	0000_0000h	37.5.7/990
80	eMIOS UC A register n (eMIOS_A3)	32	R/W	0000_0000h	37.5.1/979
84	eMIOS UC B register n (eMIOS_B3)	32	R/W	0000_0000h	37.5.2/980
88	eMIOS CNT register n (eMIOS_CNT3)	32	R/W	0000_0000h	37.5.3/981
8C	eMIOS UC Control register n (eMIOS_C3)	32	R/W	0000_0000h	37.5.4/982
90	eMIOS UC Status register n (eMIOS_S3)	32	R/W	See section	37.5.5/988
94	eMIOS Alternate Address register n (eMIOS_ALTA3)	32	R/W	0000_0000h	37.5.6/989
98	eMIOS UC Control 2 register n (eMIOS_C2_3)	32	R/W	0000_0000h	37.5.7/990
A0	eMIOS UC A register n (eMIOS_A4)	32	R/W	0000_0000h	37.5.1/979
A4	eMIOS UC B register n (eMIOS_B4)	32	R/W	0000_0000h	37.5.2/980
A8	eMIOS CNT register n (eMIOS_CNT4)	32	R/W	0000_0000h	37.5.3/981
AC	eMIOS UC Control register n (eMIOS_C4)	32	R/W	0000_0000h	37.5.4/982
B0	eMIOS UC Status register n (eMIOS_S4)	32	R/W	See section	37.5.5/988
B4	eMIOS Alternate Address register n (eMIOS_ALTA4)	32	R/W	0000_0000h	37.5.6/989
B8	eMIOS UC Control 2 register n (eMIOS_C2_4)	32	R/W	0000_0000h	37.5.7/990
C0	eMIOS UC A register n (eMIOS_A5)	32	R/W	0000_0000h	37.5.1/979
C4	eMIOS UC B register n (eMIOS_B5)	32	R/W	0000_0000h	37.5.2/980
C8	eMIOS CNT register n (eMIOS_CNT5)	32	R/W	0000_0000h	37.5.3/981
CC	eMIOS UC Control register n (eMIOS_C5)	32	R/W	0000_0000h	37.5.4/982
D0	eMIOS UC Status register n (eMIOS_S5)	32	R/W	See section	37.5.5/988
D4	eMIOS Alternate Address register n (eMIOS_ALTA5)	32	R/W	0000_0000h	37.5.6/989
D8	eMIOS UC Control 2 register n (eMIOS_C2_5)	32	R/W	0000_0000h	37.5.7/990
E0	eMIOS UC A register n (eMIOS_A6)	32	R/W	0000_0000h	37.5.1/979
E4	eMIOS UC B register n (eMIOS_B6)	32	R/W	0000_0000h	37.5.2/980
E8	eMIOS CNT register n (eMIOS_CNT6)	32	R/W	0000_0000h	37.5.3/981
EC	eMIOS UC Control register n (eMIOS_C6)	32	R/W	0000_0000h	37.5.4/982
F0	eMIOS UC Status register n (eMIOS_S6)	32	R/W	See section	37.5.5/988
F4	eMIOS Alternate Address register n (eMIOS_ALTA6)	32	R/W	0000_0000h	37.5.6/989
F8	eMIOS UC Control 2 register n (eMIOS_C2_6)	32	R/W	0000_0000h	37.5.7/990
100	eMIOS UC A register n (eMIOS_A7)	32	R/W	0000_0000h	37.5.1/979
104	eMIOS UC B register n (eMIOS_B7)	32	R/W	0000_0000h	37.5.2/980

Table continues on the next page...

eMIOS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
108	eMIOS CNT register n (eMIOS_CNT7)	32	R/W	0000_0000h	37.5.3/981
10C	eMIOS UC Control register n (eMIOS_C7)	32	R/W	0000_0000h	37.5.4/982
110	eMIOS UC Status register n (eMIOS_S7)	32	R/W	See section	37.5.5/988
114	eMIOS Alternate Address register n (eMIOS_ALTA7)	32	R/W	0000_0000h	37.5.6/989
118	eMIOS UC Control 2 register n (eMIOS_C2_7)	32	R/W	0000_0000h	37.5.7/990
120	eMIOS UC A register n (eMIOS_A8)	32	R/W	0000_0000h	37.5.1/979
124	eMIOS UC B register n (eMIOS_B8)	32	R/W	0000_0000h	37.5.2/980
128	eMIOS CNT register n (eMIOS_CNT8)	32	R/W	0000_0000h	37.5.3/981
12C	eMIOS UC Control register n (eMIOS_C8)	32	R/W	0000_0000h	37.5.4/982
130	eMIOS UC Status register n (eMIOS_S8)	32	R/W	See section	37.5.5/988
134	eMIOS Alternate Address register n (eMIOS_ALTA8)	32	R/W	0000_0000h	37.5.6/989
138	eMIOS UC Control 2 register n (eMIOS_C2_8)	32	R/W	0000_0000h	37.5.7/990
140	eMIOS UC A register n (eMIOS_A9)	32	R/W	0000_0000h	37.5.1/979
144	eMIOS UC B register n (eMIOS_B9)	32	R/W	0000_0000h	37.5.2/980
148	eMIOS CNT register n (eMIOS_CNT9)	32	R/W	0000_0000h	37.5.3/981
14C	eMIOS UC Control register n (eMIOS_C9)	32	R/W	0000_0000h	37.5.4/982
150	eMIOS UC Status register n (eMIOS_S9)	32	R/W	See section	37.5.5/988
154	eMIOS Alternate Address register n (eMIOS_ALTA9)	32	R/W	0000_0000h	37.5.6/989
158	eMIOS UC Control 2 register n (eMIOS_C2_9)	32	R/W	0000_0000h	37.5.7/990
160	eMIOS UC A register n (eMIOS_A10)	32	R/W	0000_0000h	37.5.1/979
164	eMIOS UC B register n (eMIOS_B10)	32	R/W	0000_0000h	37.5.2/980
168	eMIOS CNT register n (eMIOS_CNT10)	32	R/W	0000_0000h	37.5.3/981
16C	eMIOS UC Control register n (eMIOS_C10)	32	R/W	0000_0000h	37.5.4/982
170	eMIOS UC Status register n (eMIOS_S10)	32	R/W	See section	37.5.5/988
174	eMIOS Alternate Address register n (eMIOS_ALTA10)	32	R/W	0000_0000h	37.5.6/989
178	eMIOS UC Control 2 register n (eMIOS_C2_10)	32	R/W	0000_0000h	37.5.7/990
180	eMIOS UC A register n (eMIOS_A11)	32	R/W	0000_0000h	37.5.1/979
184	eMIOS UC B register n (eMIOS_B11)	32	R/W	0000_0000h	37.5.2/980
188	eMIOS CNT register n (eMIOS_CNT11)	32	R/W	0000_0000h	37.5.3/981
18C	eMIOS UC Control register n (eMIOS_C11)	32	R/W	0000_0000h	37.5.4/982
190	eMIOS UC Status register n (eMIOS_S11)	32	R/W	See section	37.5.5/988
194	eMIOS Alternate Address register n (eMIOS_ALTA11)	32	R/W	0000_0000h	37.5.6/989
198	eMIOS UC Control 2 register n (eMIOS_C2_11)	32	R/W	0000_0000h	37.5.7/990
1A0	eMIOS UC A register n (eMIOS_A12)	32	R/W	0000_0000h	37.5.1/979
1A4	eMIOS UC B register n (eMIOS_B12)	32	R/W	0000_0000h	37.5.2/980
1A8	eMIOS CNT register n (eMIOS_CNT12)	32	R/W	0000_0000h	37.5.3/981
1AC	eMIOS UC Control register n (eMIOS_C12)	32	R/W	0000_0000h	37.5.4/982
1B0	eMIOS UC Status register n (eMIOS_S12)	32	R/W	See section	37.5.5/988
1B4	eMIOS Alternate Address register n (eMIOS_ALTA12)	32	R/W	0000_0000h	37.5.6/989

Table continues on the next page...

eMIOS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1B8	eMIOS UC Control 2 register n (eMIOS_C2_12)	32	R/W	0000_0000h	37.5.7/990
1C0	eMIOS UC A register n (eMIOS_A13)	32	R/W	0000_0000h	37.5.1/979
1C4	eMIOS UC B register n (eMIOS_B13)	32	R/W	0000_0000h	37.5.2/980
1C8	eMIOS CNT register n (eMIOS_CNT13)	32	R/W	0000_0000h	37.5.3/981
1CC	eMIOS UC Control register n (eMIOS_C13)	32	R/W	0000_0000h	37.5.4/982
1D0	eMIOS UC Status register n (eMIOS_S13)	32	R/W	See section	37.5.5/988
1D4	eMIOS Alternate Address register n (eMIOS_ALTA13)	32	R/W	0000_0000h	37.5.6/989
1D8	eMIOS UC Control 2 register n (eMIOS_C2_13)	32	R/W	0000_0000h	37.5.7/990
1E0	eMIOS UC A register n (eMIOS_A14)	32	R/W	0000_0000h	37.5.1/979
1E4	eMIOS UC B register n (eMIOS_B14)	32	R/W	0000_0000h	37.5.2/980
1E8	eMIOS CNT register n (eMIOS_CNT14)	32	R/W	0000_0000h	37.5.3/981
1EC	eMIOS UC Control register n (eMIOS_C14)	32	R/W	0000_0000h	37.5.4/982
1F0	eMIOS UC Status register n (eMIOS_S14)	32	R/W	See section	37.5.5/988
1F4	eMIOS Alternate Address register n (eMIOS_ALTA14)	32	R/W	0000_0000h	37.5.6/989
1F8	eMIOS UC Control 2 register n (eMIOS_C2_14)	32	R/W	0000_0000h	37.5.7/990
200	eMIOS UC A register n (eMIOS_A15)	32	R/W	0000_0000h	37.5.1/979
204	eMIOS UC B register n (eMIOS_B15)	32	R/W	0000_0000h	37.5.2/980
208	eMIOS CNT register n (eMIOS_CNT15)	32	R/W	0000_0000h	37.5.3/981
20C	eMIOS UC Control register n (eMIOS_C15)	32	R/W	0000_0000h	37.5.4/982
210	eMIOS UC Status register n (eMIOS_S15)	32	R/W	See section	37.5.5/988
214	eMIOS Alternate Address register n (eMIOS_ALTA15)	32	R/W	0000_0000h	37.5.6/989
218	eMIOS UC Control 2 register n (eMIOS_C2_15)	32	R/W	0000_0000h	37.5.7/990
220	eMIOS UC A register n (eMIOS_A16)	32	R/W	0000_0000h	37.5.1/979
224	eMIOS UC B register n (eMIOS_B16)	32	R/W	0000_0000h	37.5.2/980
228	eMIOS CNT register n (eMIOS_CNT16)	32	R/W	0000_0000h	37.5.3/981
22C	eMIOS UC Control register n (eMIOS_C16)	32	R/W	0000_0000h	37.5.4/982
230	eMIOS UC Status register n (eMIOS_S16)	32	R/W	See section	37.5.5/988
234	eMIOS Alternate Address register n (eMIOS_ALTA16)	32	R/W	0000_0000h	37.5.6/989
238	eMIOS UC Control 2 register n (eMIOS_C2_16)	32	R/W	0000_0000h	37.5.7/990
240	eMIOS UC A register n (eMIOS_A17)	32	R/W	0000_0000h	37.5.1/979
244	eMIOS UC B register n (eMIOS_B17)	32	R/W	0000_0000h	37.5.2/980
248	eMIOS CNT register n (eMIOS_CNT17)	32	R/W	0000_0000h	37.5.3/981
24C	eMIOS UC Control register n (eMIOS_C17)	32	R/W	0000_0000h	37.5.4/982
250	eMIOS UC Status register n (eMIOS_S17)	32	R/W	See section	37.5.5/988
254	eMIOS Alternate Address register n (eMIOS_ALTA17)	32	R/W	0000_0000h	37.5.6/989
258	eMIOS UC Control 2 register n (eMIOS_C2_17)	32	R/W	0000_0000h	37.5.7/990
260	eMIOS UC A register n (eMIOS_A18)	32	R/W	0000_0000h	37.5.1/979
264	eMIOS UC B register n (eMIOS_B18)	32	R/W	0000_0000h	37.5.2/980
268	eMIOS CNT register n (eMIOS_CNT18)	32	R/W	0000_0000h	37.5.3/981

Table continues on the next page...

eMIOS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
26C	eMIOS UC Control register n (eMIOS_C18)	32	R/W	0000_0000h	37.5.4/982
270	eMIOS UC Status register n (eMIOS_S18)	32	R/W	See section	37.5.5/988
274	eMIOS Alternate Address register n (eMIOS_ALTA18)	32	R/W	0000_0000h	37.5.6/989
278	eMIOS UC Control 2 register n (eMIOS_C2_18)	32	R/W	0000_0000h	37.5.7/990
280	eMIOS UC A register n (eMIOS_A19)	32	R/W	0000_0000h	37.5.1/979
284	eMIOS UC B register n (eMIOS_B19)	32	R/W	0000_0000h	37.5.2/980
288	eMIOS CNT register n (eMIOS_CNT19)	32	R/W	0000_0000h	37.5.3/981
28C	eMIOS UC Control register n (eMIOS_C19)	32	R/W	0000_0000h	37.5.4/982
290	eMIOS UC Status register n (eMIOS_S19)	32	R/W	See section	37.5.5/988
294	eMIOS Alternate Address register n (eMIOS_ALTA19)	32	R/W	0000_0000h	37.5.6/989
298	eMIOS UC Control 2 register n (eMIOS_C2_19)	32	R/W	0000_0000h	37.5.7/990
2A0	eMIOS UC A register n (eMIOS_A20)	32	R/W	0000_0000h	37.5.1/979
2A4	eMIOS UC B register n (eMIOS_B20)	32	R/W	0000_0000h	37.5.2/980
2A8	eMIOS CNT register n (eMIOS_CNT20)	32	R/W	0000_0000h	37.5.3/981
2AC	eMIOS UC Control register n (eMIOS_C20)	32	R/W	0000_0000h	37.5.4/982
2B0	eMIOS UC Status register n (eMIOS_S20)	32	R/W	See section	37.5.5/988
2B4	eMIOS Alternate Address register n (eMIOS_ALTA20)	32	R/W	0000_0000h	37.5.6/989
2B8	eMIOS UC Control 2 register n (eMIOS_C2_20)	32	R/W	0000_0000h	37.5.7/990
2C0	eMIOS UC A register n (eMIOS_A21)	32	R/W	0000_0000h	37.5.1/979
2C4	eMIOS UC B register n (eMIOS_B21)	32	R/W	0000_0000h	37.5.2/980
2C8	eMIOS CNT register n (eMIOS_CNT21)	32	R/W	0000_0000h	37.5.3/981
2CC	eMIOS UC Control register n (eMIOS_C21)	32	R/W	0000_0000h	37.5.4/982
2D0	eMIOS UC Status register n (eMIOS_S21)	32	R/W	See section	37.5.5/988
2D4	eMIOS Alternate Address register n (eMIOS_ALTA21)	32	R/W	0000_0000h	37.5.6/989
2D8	eMIOS UC Control 2 register n (eMIOS_C2_21)	32	R/W	0000_0000h	37.5.7/990
2E0	eMIOS UC A register n (eMIOS_A22)	32	R/W	0000_0000h	37.5.1/979
2E4	eMIOS UC B register n (eMIOS_B22)	32	R/W	0000_0000h	37.5.2/980
2E8	eMIOS CNT register n (eMIOS_CNT22)	32	R/W	0000_0000h	37.5.3/981
2EC	eMIOS UC Control register n (eMIOS_C22)	32	R/W	0000_0000h	37.5.4/982
2F0	eMIOS UC Status register n (eMIOS_S22)	32	R/W	See section	37.5.5/988
2F4	eMIOS Alternate Address register n (eMIOS_ALTA22)	32	R/W	0000_0000h	37.5.6/989
2F8	eMIOS UC Control 2 register n (eMIOS_C2_22)	32	R/W	0000_0000h	37.5.7/990
300	eMIOS UC A register n (eMIOS_A23)	32	R/W	0000_0000h	37.5.1/979
304	eMIOS UC B register n (eMIOS_B23)	32	R/W	0000_0000h	37.5.2/980
308	eMIOS CNT register n (eMIOS_CNT23)	32	R/W	0000_0000h	37.5.3/981
30C	eMIOS UC Control register n (eMIOS_C23)	32	R/W	0000_0000h	37.5.4/982
310	eMIOS UC Status register n (eMIOS_S23)	32	R/W	See section	37.5.5/988
314	eMIOS Alternate Address register n (eMIOS_ALTA23)	32	R/W	0000_0000h	37.5.6/989
318	eMIOS UC Control 2 register n (eMIOS_C2_23)	32	R/W	0000_0000h	37.5.7/990

Table continues on the next page...

eMIOS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
320	eMIOS UC A register n (eMIOS_A24)	32	R/W	0000_0000h	37.5.1/979
324	eMIOS UC B register n (eMIOS_B24)	32	R/W	0000_0000h	37.5.2/980
328	eMIOS CNT register n (eMIOS_CNT24)	32	R/W	0000_0000h	37.5.3/981
32C	eMIOS UC Control register n (eMIOS_C24)	32	R/W	0000_0000h	37.5.4/982
330	eMIOS UC Status register n (eMIOS_S24)	32	R/W	See section	37.5.5/988
334	eMIOS Alternate Address register n (eMIOS_ALTA24)	32	R/W	0000_0000h	37.5.6/989
338	eMIOS UC Control 2 register n (eMIOS_C2_24)	32	R/W	0000_0000h	37.5.7/990
340	eMIOS UC A register n (eMIOS_A25)	32	R/W	0000_0000h	37.5.1/979
344	eMIOS UC B register n (eMIOS_B25)	32	R/W	0000_0000h	37.5.2/980
348	eMIOS CNT register n (eMIOS_CNT25)	32	R/W	0000_0000h	37.5.3/981
34C	eMIOS UC Control register n (eMIOS_C25)	32	R/W	0000_0000h	37.5.4/982
350	eMIOS UC Status register n (eMIOS_S25)	32	R/W	See section	37.5.5/988
354	eMIOS Alternate Address register n (eMIOS_ALTA25)	32	R/W	0000_0000h	37.5.6/989
358	eMIOS UC Control 2 register n (eMIOS_C2_25)	32	R/W	0000_0000h	37.5.7/990
360	eMIOS UC A register n (eMIOS_A26)	32	R/W	0000_0000h	37.5.1/979
364	eMIOS UC B register n (eMIOS_B26)	32	R/W	0000_0000h	37.5.2/980
368	eMIOS CNT register n (eMIOS_CNT26)	32	R/W	0000_0000h	37.5.3/981
36C	eMIOS UC Control register n (eMIOS_C26)	32	R/W	0000_0000h	37.5.4/982
370	eMIOS UC Status register n (eMIOS_S26)	32	R/W	See section	37.5.5/988
374	eMIOS Alternate Address register n (eMIOS_ALTA26)	32	R/W	0000_0000h	37.5.6/989
378	eMIOS UC Control 2 register n (eMIOS_C2_26)	32	R/W	0000_0000h	37.5.7/990
380	eMIOS UC A register n (eMIOS_A27)	32	R/W	0000_0000h	37.5.1/979
384	eMIOS UC B register n (eMIOS_B27)	32	R/W	0000_0000h	37.5.2/980
388	eMIOS CNT register n (eMIOS_CNT27)	32	R/W	0000_0000h	37.5.3/981
38C	eMIOS UC Control register n (eMIOS_C27)	32	R/W	0000_0000h	37.5.4/982
390	eMIOS UC Status register n (eMIOS_S27)	32	R/W	See section	37.5.5/988
394	eMIOS Alternate Address register n (eMIOS_ALTA27)	32	R/W	0000_0000h	37.5.6/989
398	eMIOS UC Control 2 register n (eMIOS_C2_27)	32	R/W	0000_0000h	37.5.7/990
3A0	eMIOS UC A register n (eMIOS_A28)	32	R/W	0000_0000h	37.5.1/979
3A4	eMIOS UC B register n (eMIOS_B28)	32	R/W	0000_0000h	37.5.2/980
3A8	eMIOS CNT register n (eMIOS_CNT28)	32	R/W	0000_0000h	37.5.3/981
3AC	eMIOS UC Control register n (eMIOS_C28)	32	R/W	0000_0000h	37.5.4/982
3B0	eMIOS UC Status register n (eMIOS_S28)	32	R/W	See section	37.5.5/988
3B4	eMIOS Alternate Address register n (eMIOS_ALTA28)	32	R/W	0000_0000h	37.5.6/989
3B8	eMIOS UC Control 2 register n (eMIOS_C2_28)	32	R/W	0000_0000h	37.5.7/990
3C0	eMIOS UC A register n (eMIOS_A29)	32	R/W	0000_0000h	37.5.1/979
3C4	eMIOS UC B register n (eMIOS_B29)	32	R/W	0000_0000h	37.5.2/980
3C8	eMIOS CNT register n (eMIOS_CNT29)	32	R/W	0000_0000h	37.5.3/981
3CC	eMIOS UC Control register n (eMIOS_C29)	32	R/W	0000_0000h	37.5.4/982

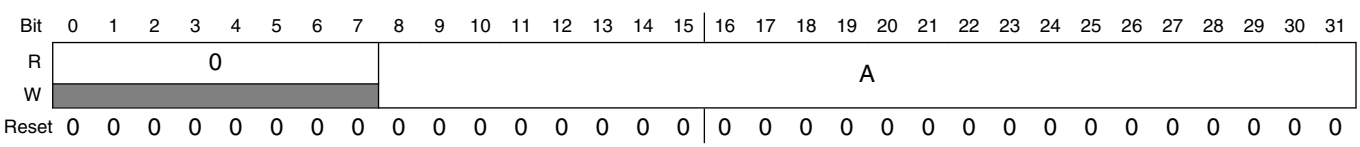
Table continues on the next page...

eMIOS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3D0	eMIOS UC Status register n (eMIOS_S29)	32	R/W	See section	37.5.5/988
3D4	eMIOS Alternate Address register n (eMIOS_ALTA29)	32	R/W	0000_0000h	37.5.6/989
3D8	eMIOS UC Control 2 register n (eMIOS_C2_29)	32	R/W	0000_0000h	37.5.7/990
3E0	eMIOS UC A register n (eMIOS_A30)	32	R/W	0000_0000h	37.5.1/979
3E4	eMIOS UC B register n (eMIOS_B30)	32	R/W	0000_0000h	37.5.2/980
3E8	eMIOS CNT register n (eMIOS_CNT30)	32	R/W	0000_0000h	37.5.3/981
3EC	eMIOS UC Control register n (eMIOS_C30)	32	R/W	0000_0000h	37.5.4/982
3F0	eMIOS UC Status register n (eMIOS_S30)	32	R/W	See section	37.5.5/988
3F4	eMIOS Alternate Address register n (eMIOS_ALTA30)	32	R/W	0000_0000h	37.5.6/989
3F8	eMIOS UC Control 2 register n (eMIOS_C2_30)	32	R/W	0000_0000h	37.5.7/990
400	eMIOS UC A register n (eMIOS_A31)	32	R/W	0000_0000h	37.5.1/979
404	eMIOS UC B register n (eMIOS_B31)	32	R/W	0000_0000h	37.5.2/980
408	eMIOS CNT register n (eMIOS_CNT31)	32	R/W	0000_0000h	37.5.3/981
40C	eMIOS UC Control register n (eMIOS_C31)	32	R/W	0000_0000h	37.5.4/982
410	eMIOS UC Status register n (eMIOS_S31)	32	R/W	See section	37.5.5/988
414	eMIOS Alternate Address register n (eMIOS_ALTA31)	32	R/W	0000_0000h	37.5.6/989
418	eMIOS UC Control 2 register n (eMIOS_C2_31)	32	R/W	0000_0000h	37.5.7/990

37.5.1 eMIOS UC A register n (eMIOS_A*n*)

Address: 0h base + 20h offset + (32d × i), where i=0d to 31d



eMIOS_A*n* field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 A	Depending on the mode of operation, internal registers A1[0:23] or A2[0:23], used for matches and captures, can be assigned to address EMIOS_An. Both A1 and A2 are cleared by reset. The table in the EMIOS_Bn register description summarizes the EMIOS_An writing and reading accesses for all operation modes. Values not listed are reserved. For more information, see the “UC Modes of Operation” section.

37.5.2 eMIOS UC B register n (eMIOS_B*n*)

Depending on the mode of operation, internal registers B1[0:23] or B2[0:23] can be assigned to address EMIOS_Bn. Both B1 and B2 are cleared by reset. The table below summarizes the EMIOS_Bn writing and reading accesses for all operation modes. Values not listed are reserved. For more information see the “UC Modes of Operation” section.

Depending on the channel configuration, it may have an EMIOS_B register or not. EMIOS_B register is required for the following modes: DAOC, IPM, IPWM, OPWMB, OPWMT, OPWFMB, OPWMCB, MC, MCB.

The following table summarizes the EMIOS_An, EMIOS_Bn, and EMIOS_ALTAn writing and reading accesses for all operation modes. Values not listed are reserved. For more information, see the “UC Modes of Operation” section.

Table 37-9. EMIOS_An, EMIOS_Bn and EMIOS_ALTAn values assignment

Operation mode	Register access					
	Write	Read	Write	Read	Alt write	Alt read
GPIO	A1, A2	A1	B1, B2	B1	A2	A2
SAIC[1]	–	A2	B2	B2	–	–
SAOC[1]	A2	A1	B2	B2	–	–
IPWM	–	A2	–	B1	–	–
IPM	–	A2	–	B1	–	–
DAOC	A2	A1	B2	B1	–	–
PEA	A1	A2	–	B1	–	–
PEC[1]	A1	A1	B1	B1	–	A2
WPTA	A1	A1	B1	B1	–	A2
MC[1]	A2	A1	B2	B2	–	–
OPWFM	A2	A1	B2	B1	–	–
OPWMC	A2	A1	B2	B1	–	–
OPWM	A2	A1	B2	B1	–	–
OPWMT	A2	A1	B2	B1	A2	A2
MCB[1]	A2	A1	B2	B1	–	–
OPWFMB	A2	A1	B2	B1	–	–
OPWMCB	A2	A1	B2	B1	–	–
OPWMB	A2	A1	B2	B1	–	–

1. In these modes, the register EMIOS_Bn is not used, but B2 can be accessed.

Address: 0h base + 24h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								B																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_B*n* field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 B	See text above register figure.

37.5.3 eMIOS CNT register n (eMIOS_CNT*n*)

When GPIO mode is selected or the channel is frozen, the EMIOS_CNTn register is read/write. For all others modes, the EMIOS_CNTn is a read-only register. When entering some operation modes, this register is automatically cleared (refer to the “UC Modes of Operation” section for details).

EMIOS_CNTn register is required for the following modes:

- OPWFMB
- OPWMCB
- MC
- MCB

For other modes, the internal counter is not used. Therefore, depending on the modes available in a channel, the internal counter may be absent.

Address: 0h base + 28h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																C															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_CNT*n* field descriptions

Field	Description
0–15 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
16–31 C	The C field contains the value of the internal counter.

37.5.4 eMIOS UC Control register n (eMIOS_C*n*)

The UC Control register gathers several read/write control bits affecting the UC operation conditions.

NOTE

This register does not include a FORCMB bit for Channel Types A and C.

Address: 0h base + 2Ch offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FREN	ODIS	ODISSL		UCPRE		UCPREN	DMA	0	IF				FCK	FEN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				0	BSL		EDSEL	EDPOL	MODE						
W			FORCMA	FORCMB												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eMIOS_C*n* field descriptions

Field	Description
0 FREN	Freeze Enable bit. The FREN bit, if set and validated by FRZ bit in EMIOS_MCR register allows the channel to enter freeze state, freezing all registers values when in debug mode and allowing the MCU to perform debug functions. 0 Normal operation 1 Freeze UC registers values
1 ODIS	Output Disable bit. The ODIS bit allows disabling the output pin when running any of the output modes with the exception of GPIO mode.

Table continues on the next page...

eMIOS_C*n* field descriptions (continued)

Field	Description
	0 The output pin operates normally 1 If the selected Output Disable Input signal is asserted, the output pin goes to EDPOL for OPWFMB and OPWMB modes and to the complement of EDPOL for other output modes, but the Unified Channel continues to operate normally, i.e., it continues to produce FLAG and matches. When the selected Output Disable Input signal is negated, the output pin operates normally.
2–3 ODISSL	Output Disable select bits. The ODISSL[0:1] bits select one of the four output disable input signals, as shown in the following table. **Table 37-10. UC ODISSL selection** ODISSL[0:1] / input signal: 00 Output Disable Input 0; 01 Output Disable Input 1; 10 Output Disable Input 2; 11 Output Disable Input 3
4–5 UCPRE	Prescaler bits. The UCPRE[0:1] bits select the clock divider value for the internal prescaler of Unified Channel, as shown in the following table. **Table 37-11. UC Internal Prescaler clock divider** UCPRE[0:1] / Divide ratio: 00 1; 01 2; 10 3; 11 4 **NOTE:** The UCPRE bits are mirrored in the two LSBs of the UCEXTPRE field in register EMIOS_Cn. Any write operations to UCPRE bits will affect the respective bits at EMIOS_C2n register.
6 UCPREN	Prescaler Enable bit. The UCPREN bit enables the prescaler counter. 0 Prescaler disabled (no clock) 1 Prescaler enabled
7 DMA	Direct Memory Access bit. The DMA bit selects if FLAG generation will be used as an interrupt or as a DMA request. 0 Flag/overrun assigned to Interrupt request. 1 Flag/overrun assigned to DMA request.
8 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

eMIOS_C*n* field descriptions (continued)

Field	Description
9–12 IF	Input Filter bits. The IF[0:3] bits control the programmable input filter, selecting the minimum input pulse width that can pass through the filter, as shown in the following table. For output modes, these bits have no meaning. **Table 37-12. UC Input Filter bits** **IF[0:3]** [1] / **Minimum input Pulse width [filter clock periods]** 0000 — bypassed[2] 0001 — 02 0010 — 04 0100 — 08 1000 — 16 All others — Reserved
13 FCK	Filter Clock select bit. The FCK bit selects the clock source for the programmable input filter. 0 Prescaled clock 1 eMIOS module clock
14 FEN	FLAG Enable bit. The FEN bit allows the Unified Channel FLAG bit to generate an interrupt signal or a DMA request signal (the type of signal to be generated is defined by the DMA bit). 0 Disable (FLAG does not generate an interrupt request) 1 Enable (FLAG generates an interrupt request)
15–17 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
18 FORCMA	Force Match A bit. For output modes, the FORCMA bit is equivalent to a successful comparison on comparator A (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator A, otherwise it has no effect. For input modes, the FORCMA bit is not used and writing to it has no effect. 0 Has no effect. 1 Force a match at comparator A
19 FORCMB	Force Match B bit. For output modes, the FORCMB bit is equivalent to a successful comparison on comparator B (except that the FLAG bit is not set). This bit is cleared by reset and is always read as zero. This bit is valid for every output operation mode which uses comparator B, otherwise it has no effect. 0 Has not effect 1 Force a match at comparator B
20 Reserved	Reserved.

Table continues on the next page...

eMIOS_C*n* field descriptions (continued)

Field	Description
	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 BSL	Bus Select bits. The BSL[0:1] bits are used to select either one of the counter buses or the internal counter to be used by the Unified Channel , as shown in the following tables.

Table 37-13. UC BSL bits

BSL[0:1]	Selected bus[3]
00	All channels: counter bus[A]
01	Channels 0 to 7: counter bus[B]
	Channels 8 to 15: counter bus[C]
	Channels 16 to 23: counter bus[D]
	Channels 24 to 31: counter bus[E]
10	A system specified channel: counter bus F
11	All channels: internal counter

23 EDSEL	Edge Selection bit. For input modes, the EDSEL bit selects whether the internal counter is triggered by both edges of a pulse or just by a single edge as defined by the EDPOL bit . When not shown in the mode of operation description, this bit has no effect. 0 – Single edge triggering defined by the EDPOL bit 1 – Both edges triggering{mts_ipi_slv_saic} For GPIO in mode, the EDSEL bit selects if a FLAG can be generated. 0 – A FLAG is generated as defined by the EDPOL bit 1 – No FLAG is generated For SAOC mode, the EDSEL bit selects the behavior of the output flip-flop at each match. 0 – The EDPOL value is transferred to the output flip-flop 1 – The output flip-flop is toggled
24 EDPOL	Edge Polarity bit. For input modes, the EDPOL bit asserts which edge triggers either the internal counter or an input capture or a FLAG . When not shown in the mode of operation description, this bit has no effect. 0 – Trigger on a falling edge 1 – Trigger on a rising edge For output modes, the EDPOL bit is used to select the logic level on the output pin . 0 – A match on comparator A clears the output flip-flop, while a match on comparator B sets it 1 – A match on comparator A sets the output flip-flop, while a match on comparator B clears it
25–31 MODE	Mode selection bits. The MODE bits select the mode of operation of the Unified Channel, as shown in the following table. Values not listed are reserved.

Table continues on the next page...

eMIOS_C*n* field descriptions

Field	Description

NOTE: If a reserved value is written to the mode bit field, the results are unpredictable.

Table 37-14. UC MODE bits

MODE[0:6]	Mode of operation[4]
0000000	General purpose Input/Output mode (input)
0000001	General purpose Input/Output mode (output)
0000010	Single Action Input Capture
0000011	Single Action Output Compare
0000100	Input Pulse Width Measurement
0000101	Input Period Measurement
0000110	Double Action Output compare (with FLAG set on B match)
0000111	Double Action Output compare (with FLAG set on both match)
0001000	Pulse/Edge Accumulation (continuous)
0001001	Pulse/Edge Accumulation (single shot)
0001010	Pulse/Edge Counting (continuous)
0001011	Pulse/Edge Counting (single shot)
0001100	Quadrature Decode (for count and direction encoders type)
0001101	Quadrature Decode (for phase_A and phase_B encoders type)
0001110	Windowed Programmable Time Accumulation
0001111	Reserved
001000b[5]	Modulus Counter (Up counter with clear on match start)
001001b	Modulus Counter (Up counter with clear on match end)
00101bb[5]	Modulus Counter (Up/Down counter)
00110b0[5]	Output Pulse Width and Frequency Modulation (immediate update)
00110b1[5]	Output Pulse Width and Frequency Modulation (next period update)
00111b0[5]	Center Aligned Output Pulse Width Modulation (with trail edge dead-time)
00111b1[5]	Center Aligned Output Pulse Width Modulation (with lead edge dead-time)
01000b0[5]	Output Pulse Width Modulation (immediate update)
01000b1[5]	Output Pulse Width Modulation (next period update)
0100100 through 0100101	Reserved
0100110	Output Pulse Width Modulation with Trigger
0100111 through 1001111	Reserved
101000b[5]	Modulus Counter Buffered (Up counter)
101001b[5]	Reserved
10101bb[5]	Modulus Counter Buffered (Up/Down counter)
10110b0[5]	Output Pulse Width and Frequency Modulation Buffered

eMIOS_C*n* field descriptions (continued)

Field	Description

Table 37-14. UC MODE bits (continued)

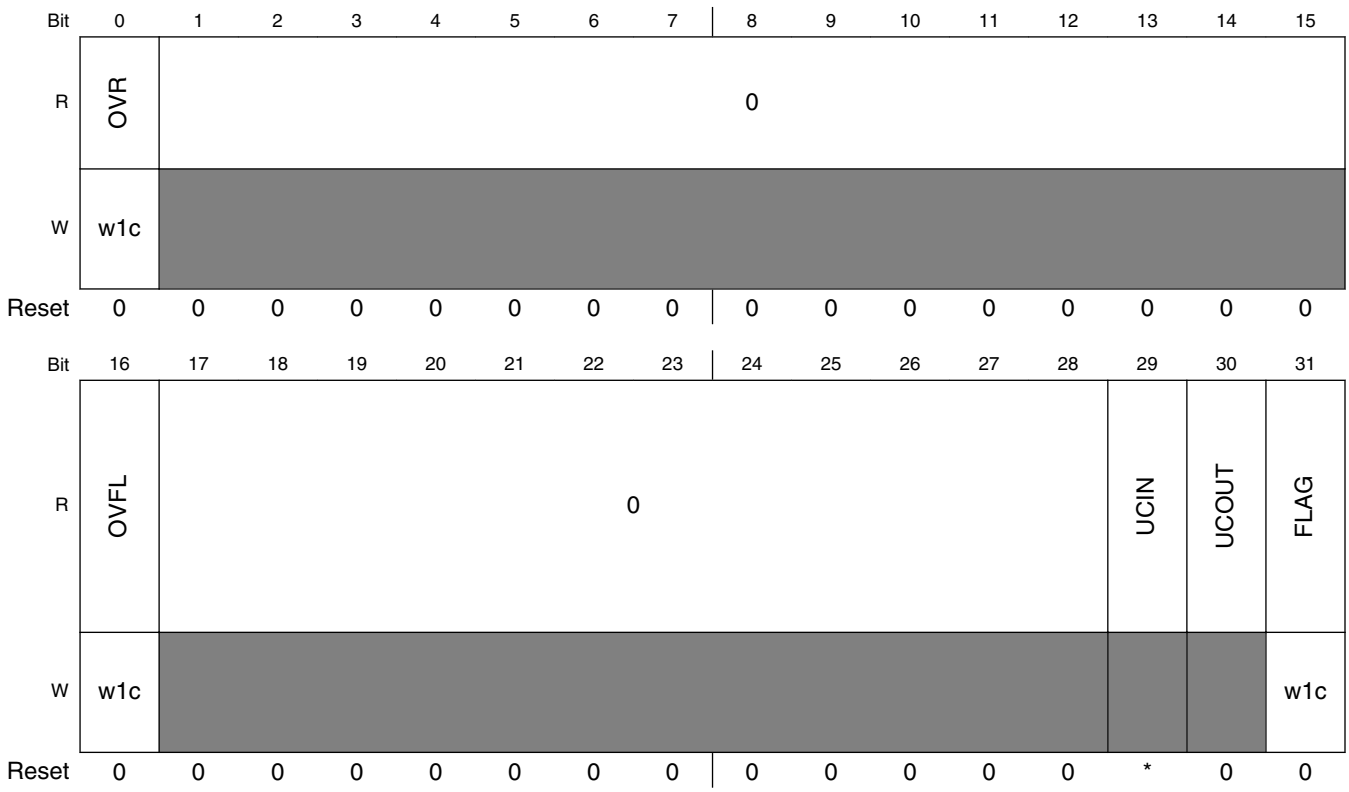
MODE[0:6]	Mode of operation[4]
10110b1[5]	Reserved
10111b0[5]	Center Aligned Output Pulse Width Modulation Buffered (with trail edge dead-time)
10111b1[5]	Center Aligned Output Pulse Width Modulation Buffered (with lead edge dead-time)
11000b0[5]	Output Pulse Width Modulation Buffered
1100001 through 1111111	Reserved

1. Filter latency (the difference in time between the input and the response) is 3 clock edges.
2. The input signal is synchronized before arriving to the digital filter.
3. Not all devices support all counter buses. See the Device Configuration chapter for details.
4. Not all modes of operation are supported. See the configuration information for details.
5. b = adjust parameters for the mode of operation. Refer to “UC Modes of Operation” section for details.

37.5.5 eMIOS UC Status register n (eMIOS_S*n*)

The UC Status register EMIOS_Sn gathers bits reflecting the status of the UC input/ output signals and the overflow condition of the internal counter.

Address: 0h base + 30h offset + (32d × i), where i=0d to 31d



* Notes:
- UCIN field: Reset value is undefined.

eMIOS_S*n* field descriptions

Field	Description
0 OVR	Overrun bit. The OVR bit indicates that FLAG generation occurred when the FLAG bit was already set . The OVR bit can be cleared either by clearing the FLAG bit or by writing a 1 to the OVR bit. 0 Overrun has not occurred 1 Overrun has occurred
1–15 Reserved	Reserved. This field is reserved.

Table continues on the next page...

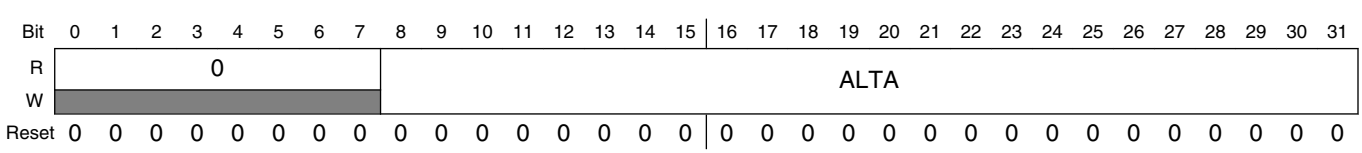
eMIOS_S*n* field descriptions (continued)

Field	Description
	This read-only field is reserved and always has the value 0.
16 OVFL	Overflow bit. The OVFL bit indicates that an overflow has occurred in the internal counter. The OVFL bit must be cleared by software writing a 1 to it. 0 No overflow 1 An overflow had occurred
17–28 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29 UCIN	Unified Channel Input pin bit. The UCIN bit reflects the input pin state after being filtered and synchronized.
30 UCOUT	Unified Channel Output pin bit. The UCOUT bit reflects the output pin state .
31 FLAG	FLAG bit. The FLAG bit is set when an input capture or a match event in the comparators occurred. The FLAG bit must be cleared by writing a 1 to it. 0 FLAG cleared 1 FLAG set event has occurred

37.5.6 eMIOS Alternate Address register n (eMIOS_ALTA*n*)

The eMIOS_ALTAn register provides an alternate address to access A2 channel registers in restricted modes (GPIO, OPWMT)only. In modes where the function of eMIOS_ALTAn register is not specified, any write access is ignored and any read access returns unspecified data. If eMIOS_An register is used along with eMIOS_ALTAn, both A1 and A2 registers can be accessed in these modes. The table in the eMIOS_Bn register's description summarizes the EMIOS_ALTAn writing and reading accesses for all EMIOS_ALTAn operation modes. See the “General purpose Input/Output mode (GPIO) Mode” section, and the “Output Pulse Width Modulation with Trigger (OPWMT) Mode” section for a more detailed description of the use of eMIOS_ALTAn register.

Address: 0h base + 34h offset + (32d × i), where i=0d to 31d

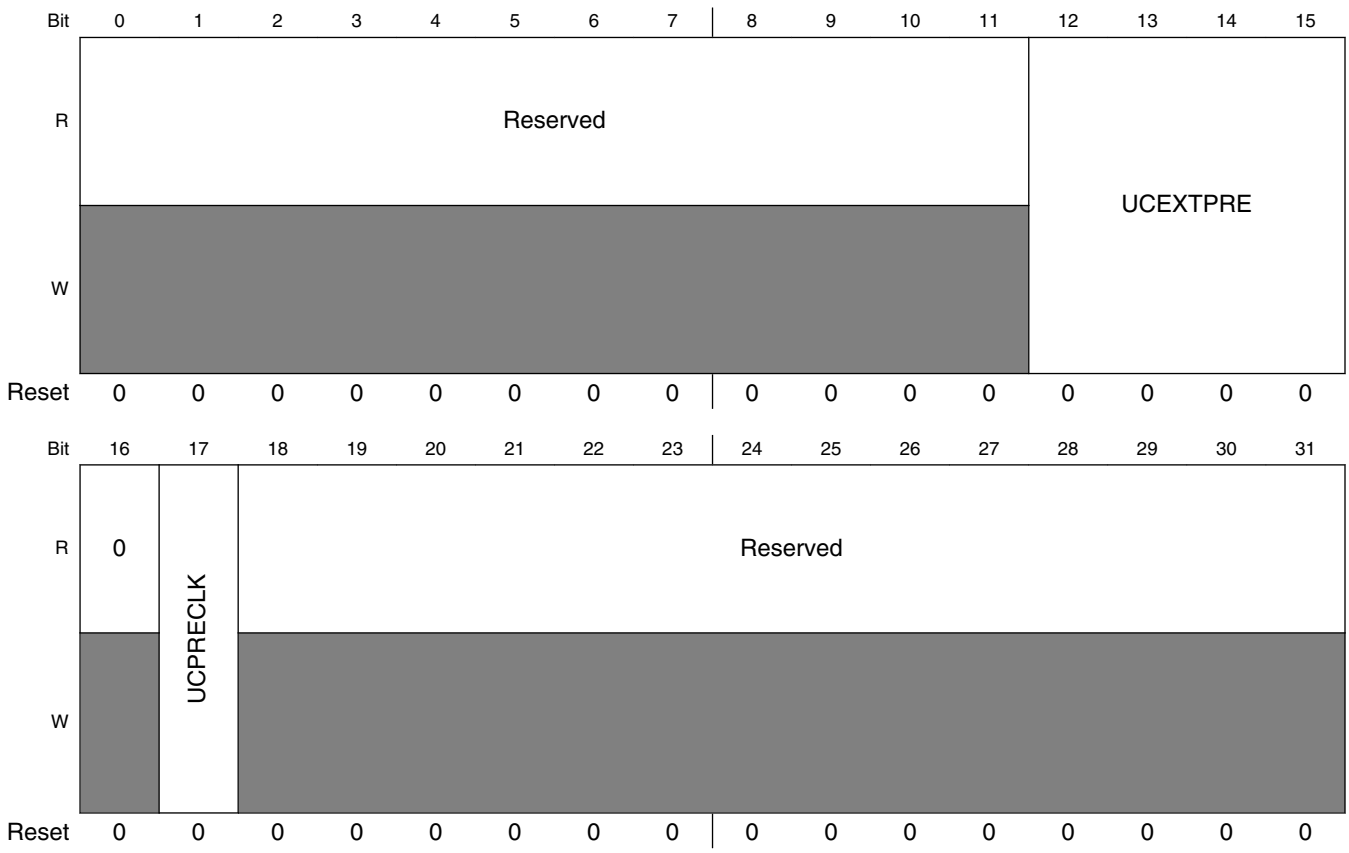


eMIOS_ALTA*n* field descriptions

Field	Description
0–7 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
8–31 ALTA	See text above register figure.

37.5.7 eMIOS UC Control 2 register n (eMIOS_C2_*n*)

Address: 0h base + 38h offset + (32d × i), where i=0d to 31d



eMIOS_C2_*n* field descriptions

Field	Description
0–11 Reserved	Reserved. This field is reserved.
12–15 UCEXTPRE	Extended Prescaler bits

Table continues on the next page...

eMIOS_C2_*n* field descriptions (continued)

Field	Description
	The UCEXTPRE bits allow the internal prescaler to be extended to 4 bits. The UCEXTPRE bits select the clock divider value for the internal prescaler of Unified Channel, as shown in the following table. **Table 37-15. UC Extended Internal Prescaler clock divider** UCEXTPRE[0:3] / Divide ratio 0000 / 1 0001 / 2 ------ / ------ 1110 / 15 1111 / 16 **NOTE:** The two LSBs of UCEXTPRE are mirrored in UCPRE bits (EMIOSC[4:5]). Any write operations to UCEXTPRE bits will affect the respective bits in the eMIOS_Cn register.
16 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
17 UCPRECLK	Prescaler Clock select bit. The UCPRECLK bit selects the clock source for the internal prescaler, 0 Prescaled Clock 1 eMIOS module clock
18–31 Reserved	Reserved. This field is reserved.

37.6 Functional Description

The eMIOS provides independent channels (Unified) that can be configured and accessed by a host MCU. Up to six time bases can be shared by the channels.

The eMIOS module is based on a multi-bus timer architecture in which several timer channels are used to drive counter buses that are shared among the channels. Up to five timebases can be shared by the channels through five counter buses (counter bus A or F, B, C, D and E) and each channel can generate its own timebase. There are six counter buses in the module: one global counter bus, shared by all channels and selectable among time bases A or F, and four local counter buses (B, C, D, and E), each one dedicated to a slice of 8 channels. Counter buses A and F are referred to as the global counter buses. Counter buses B, C, D, and E are the local counter buses.

The eMIOS counter buses are driven by channels in specific locations. Global counter bus A is driven by the channel in channel slot [23] and counter bus F is driven by the channel in channel slot [22].. Counter buses B, C, D, and E are driven by channels in slots 0, 8, 16, and 24, respectively. Counter bus A or F drives all channels. Counter bus B drives channels in slots from 0 through 7. Counter bus C drives channels in slots from 8 through 15. Counter bus D drives channels in slots from 16 through 23. Counter bus E drives channels from 24 through 31.

The eMIOS block is reset asynchronously. All registers are cleared on reset.

The following figure shows an eMIOS block configured with a mix of Unified Channels. In this configuration example, each one of the channels can be driven by the respective local bus—B for the [7:0] slice and D for the [23:16] slice—or by the global counter bus A. Note that there are Unified Channels with no external pin associated. These channels are used to trigger internal SoC signals such as interrupt requests or DMA transfer requests. Since the channel flags are also connected to the block top-level pins, they can be used as trigger signals for other blocks at chip level.

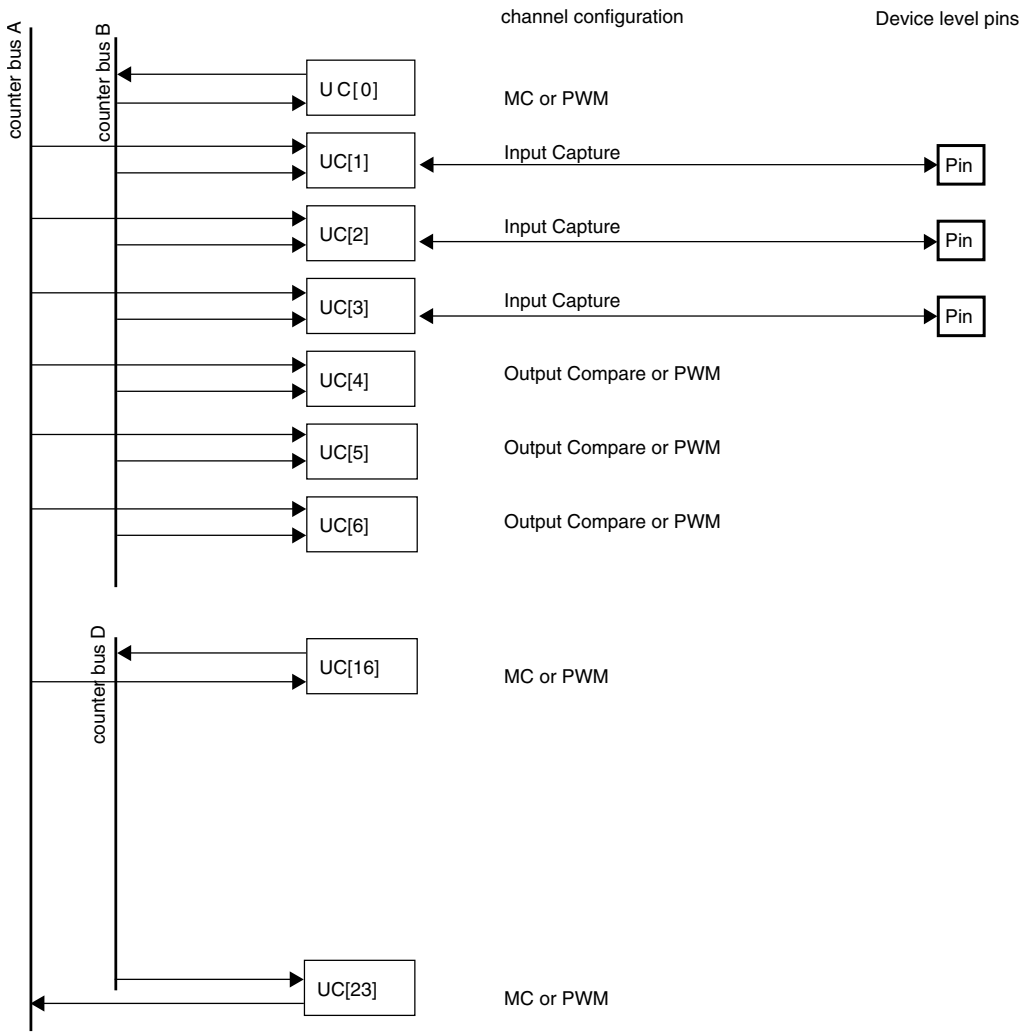


Figure 37-4. eMIOS example configuration using Unified Channels

37.6.1 Unified Channel (UC)

shows the Unified Channel block diagram. Each Unified Channel consists of the following:

- A counter bus selector, which selects the timebase to be used by the channel for all timing functions
- A programmable clock prescaler
- Two double-buffered data registers, A and B, that allow up to two input capture and/or output compare events to occur before software intervention is needed
- Two comparators (equal only), A and B, that compare the selected counter bus with the value in the data registers

- An internal counter, which can be used as a local time base or to count input events
- Programmable input filter, which ensures that only valid pin transitions are received by channel
- Programmable input edge detector, which detects the rising, falling, or either edge
- An output flip-flop, which holds the logic level to be applied to the output pin
- eMIOS Status and Control register
- An Output Disable Input selector, which selects the Output Disable Input signal that will be used as output disable

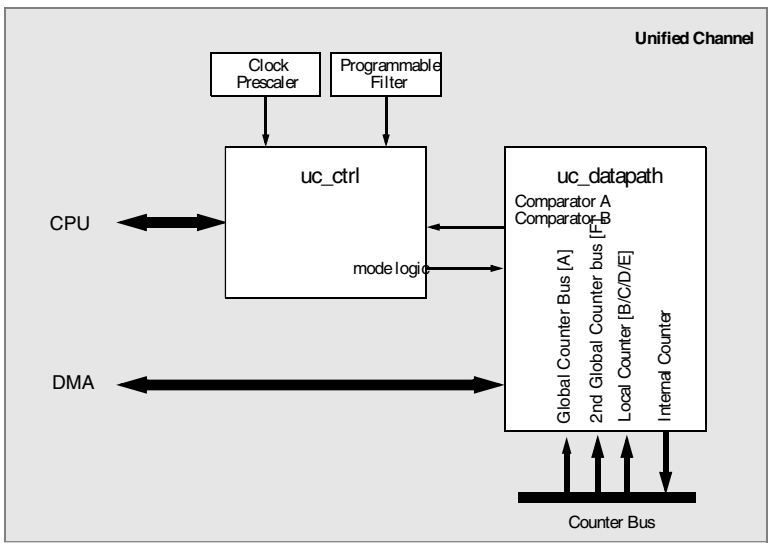


Figure 37-5. Unified Channel block diagram

shows both the Unified Channel control and datapath block diagrams. The control sub-block is responsible for the generation of signals to control the multiplexers in the datapath sub-block. Each mode is implemented by a dedicated logic independent from others modes, thus allowing to optimize the logic by disabling the mode and therefore its associated logic. Targeting the logic optimization, a set of registers is shared by the modes thus providing sequential events to be stored.

The datapath block provides the channel A and B registers, the internal timebase and comparators. Multiplexers select the input of comparators and data for the registers inputs, thus configuring the datapath to implement the channel modes. The outputs of the A and B comparators are connected to the uc_ctrl control block.

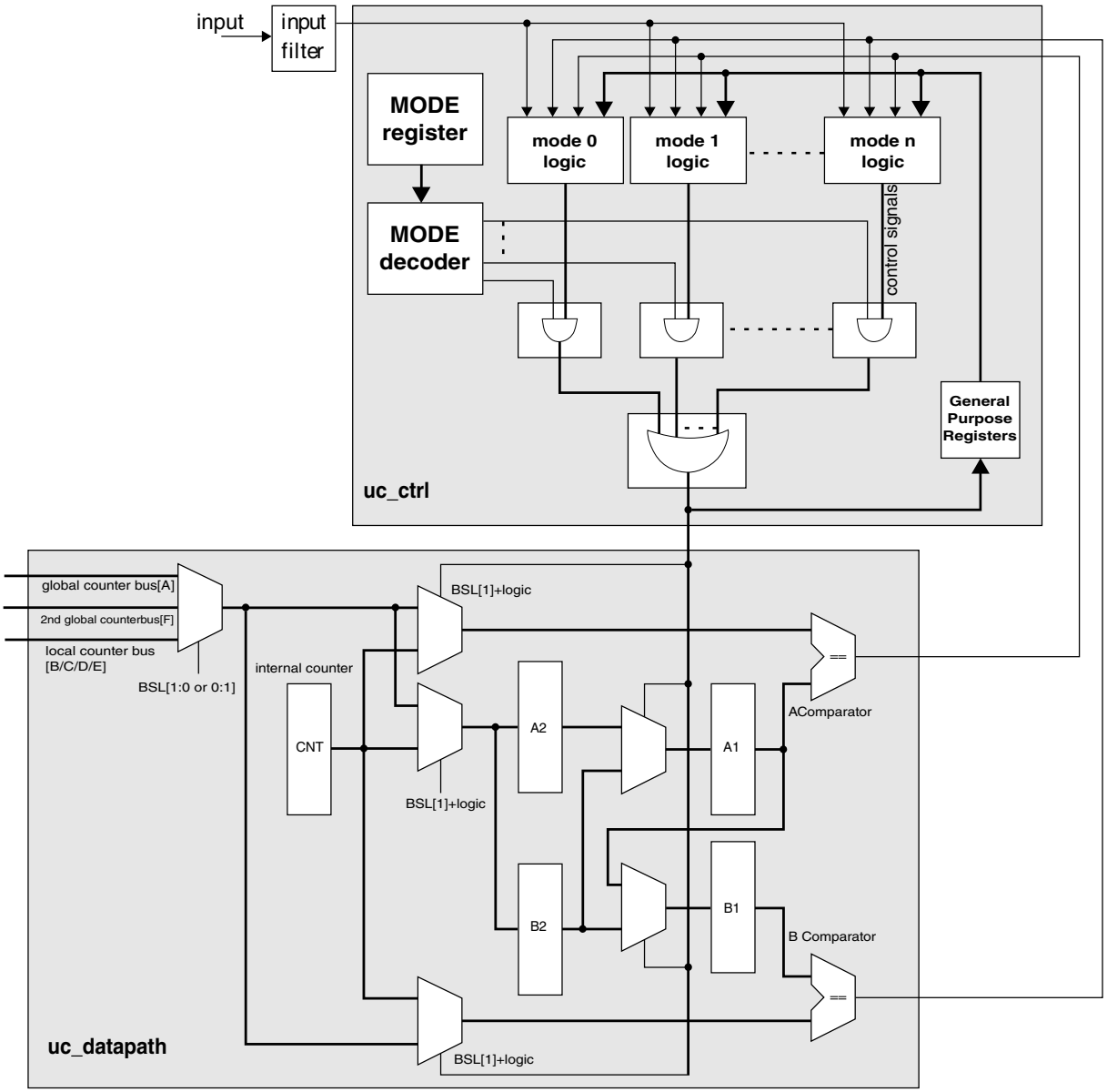


Figure 37-6. Unified Channel control and datapath block diagrams (BSL[0:1]

37.6.1.1 Unified Channel modes of operation

The mode of operation of the Unified Channel is determined by the mode select bits MODE[0:6] in the EMIOS_C_n register.

When entering an output mode (except for GPIO mode), the output flip-flop is set to the disabled state according to EMIOSCn[ODIS].

As the internal counter, EMIOSCNTn, runs in all modes (except for GPIO mode) and can be used as a timebase if the resource is not used in the current mode.

In order to provide smooth waveform generation even when A and B registers are immediately changed, it is available with the MCB, OPWFMB, OPWMB and OPWFMB modes. In these modes, A and B registers are double buffered. They are presented in separate sections because there are several basic differences between these and each corresponding nonbuffered mode.

37.6.1.1.1 General-Purpose Input/Output (GPIO) mode

In GPIO mode, all input capture and output compare functions of the Unified Channel are disabled, and the internal counter (EMIOSCNTn register) is cleared and disabled. All control bits remain accessible. In order to prepare the UC for a new operation mode, writing to registers EMIOSAn or EMIOSBn stores the same value in registers A1/A2 or B1/B2, respectively. Writing to the EMIOSALTAn register stores a value only in register A2.

MODE[6] bit selects between input (MODE[6] = 0) and output (MODE[6] = 1) modes.

When changing MODE[0:6], the application software must first enter GPIO mode to reset the Unified Channel's internal functions properly. Failure to do this could lead to invalid and unexpected output compare or input capture results or the FLAGs being set incorrectly.

In GPIO input mode (MODE[0:6] = 0000000), FLAG generation is determined according to EDPOL and EDSEL bits and the input pin status can be determined by reading the UCIN bit.

In GPIO output mode (MODE[0:6] = 0000001), the Unified Channel is used as a single output port pin and the value of the EDPOL bit is permanently transferred to the output flip-flop.

37.6.1.1.2 Single Action Input Capture (SAIC) Mode

In SAIC mode (MODE[0:6]=0000010), when a triggering event occurs on the input pin, the value on the selected time base is captured into register A2. The FLAG bit is set along with the capture event to indicate that an input capture has occurred. Register EMIOSA[n] returns the value of register A2. As soon as the SAIC mode is entered coming out from GPIO mode the channel is ready to capture events. The events are captured as soon as they occur thus reading register A always returns the value of the latest captured event. Subsequent captures are enabled with no need of further reads from EMIOSA[n] register. The FLAG is set at any time a new event is captured.

The input capture is triggered by a rising, falling or either edges in the input pin, as configured by EDPOL and EDSEL bits in EMIOSC[n] register.

Figure 37-7 and Figure 37-8 show how the Unified Channel can be used for input capture.

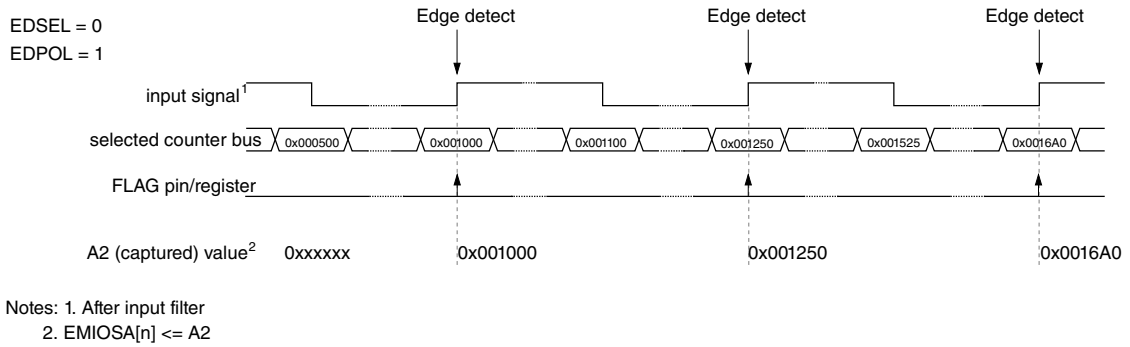


Figure 37-7. Single Action Input Capture with rising edge triggering example

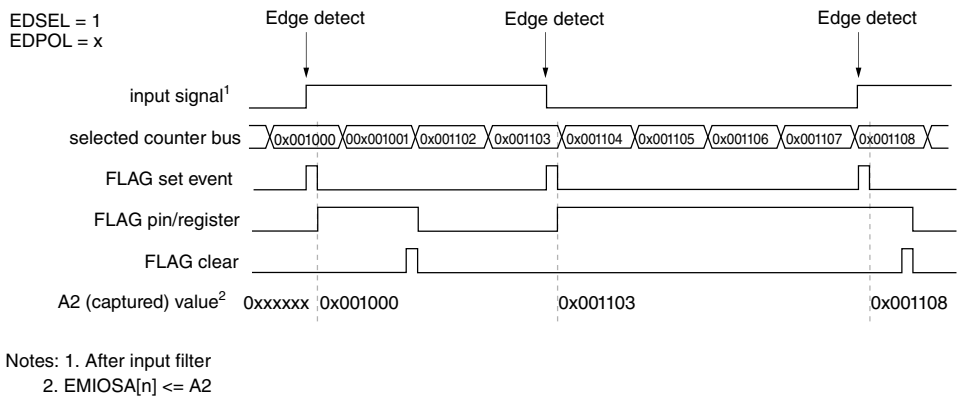


Figure 37-8. Single Action Input Capture with both edges triggering example

37.6.1.1.3 Single Action Output Compare (SAOC) Mode

In SAOC mode (MODE[0:6]=0000011) a match value is loaded in register A2 and then immediately transferred to register A1 to be compared with the selected time base. When a match occurs, the EDSEL bit selects whether the output flip-flop is toggled or the value in EDPOL is transferred to it. Along with the match the FLAG bit is set to indicate that the output compare match has occurred. Writing to register EMIOSA[n] stores the value in register A2 and reading to register EMIOSA[n] returns the value of register A1.

An output compare match can be simulated in software by setting the FORCMA bit in EMIOSC[n] register. In this case, the FLAG bit is not set.

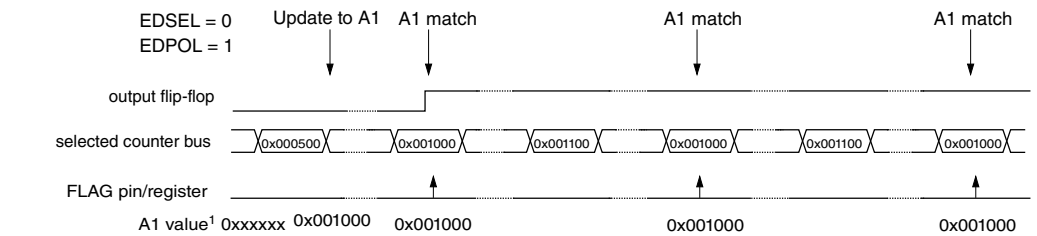
When SAOC mode is entered coming out from GPIO mode, the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

Counter bus can be either internal or external and is selected through BSL[0:1] bits.

Figure 37-9 and Figure 37-10 show how the Unified Channel can be used to perform a single output compare with EDPOL value being transferred to the output flip-flop and toggling the output flip-flop at each match, respectively. Note that once in SAOC mode the matches are enabled thus the desired match value on register A1 must be written before the mode is entered. A1 register can be updated at any time thus modifying the match value which will reflect in the output signal generated by the channel. Subsequent matches are enabled with no need of further writes to EMIOSA[n] register. The FLAG is set at the same time a match occurs (see Figure 37-11).

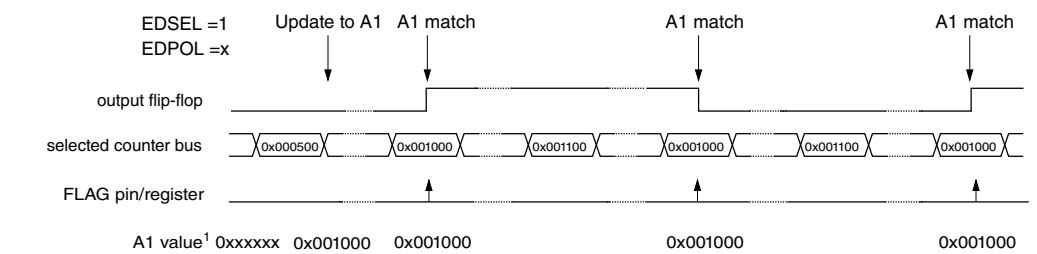
Note

The channel internal counter in SAOC mode is free-running. It starts counting as soon as the SAOC mode is entered.



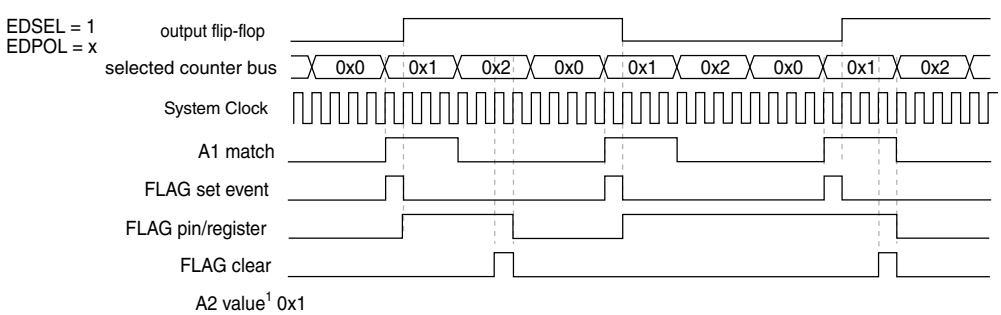
Notes: 1. EMIOSA[n] = A2
A2 = A1 according to OU[n] bit

Figure 37-9. SAOC example with EDPOL value being transferred to the output flip-flop



Notes: 1. EMIOSA[n] = A2
A2 = A1 according to OU[n] bit

Figure 37-10. SAOC example toggling the output flip-flop



Note: 1. EMIOSA[n] <= A2

Figure 37-11. SAOC example with flag behavior

37.6.1.1.4 Input Pulse Width Measurement (IPWM) Mode

The IPWM mode (MODE[0:6]=0000100) allows the measurement of the width of a positive or negative pulse by capturing the leading edge on register B1 and the trailing edge on register A2. Successive captures are done on consecutive edges of opposite polarity. The leading edge sensitivity (i.e., pulse polarity) is selected by EDPOL bit in the EMIOSC[n] register. Registers EMIOSA[n] and EMIOSB[n] return the values in register A2 and B1, respectively.

The capture function of register A2 remains disabled until the first leading edge triggers the first input capture on register B2. When this leading edge is detected, the count value of the selected time base is latched into register B2; the FLAG bit is not set. When the trailing edge is detected, the count value of the selected time base is latched into register A2 and, at the same time, the FLAG bit is set and the content of register B2 is transferred to register B1 and to register A1.

If subsequent input capture events occur while the corresponding FLAG bit is set, registers A2, B1 and A1 will be updated with the latest captured values and the FLAG will remain set. Registers EMIOSA[n] and EMIOSB[n] return the value in registers A2 and B1, respectively.

In order to guarantee coherent access, reading EMIOSA[n] forces B1 be updated with the content of register A1. At the same time transfers between B2 and B1 are disabled until the next read of EMIOSB[n] register. Reading EMIOSB[n] register forces B1 be updated with A1 register content and re-enables transfers from B2 to B1, to take effect at the next trailing edge capture. Transfers from B2 to A1 are not blocked at any time.

The input pulse width is calculated by subtracting the value in B1 from A2.

The following figure shows how the Unified Channel can be used for input pulse width measurement.

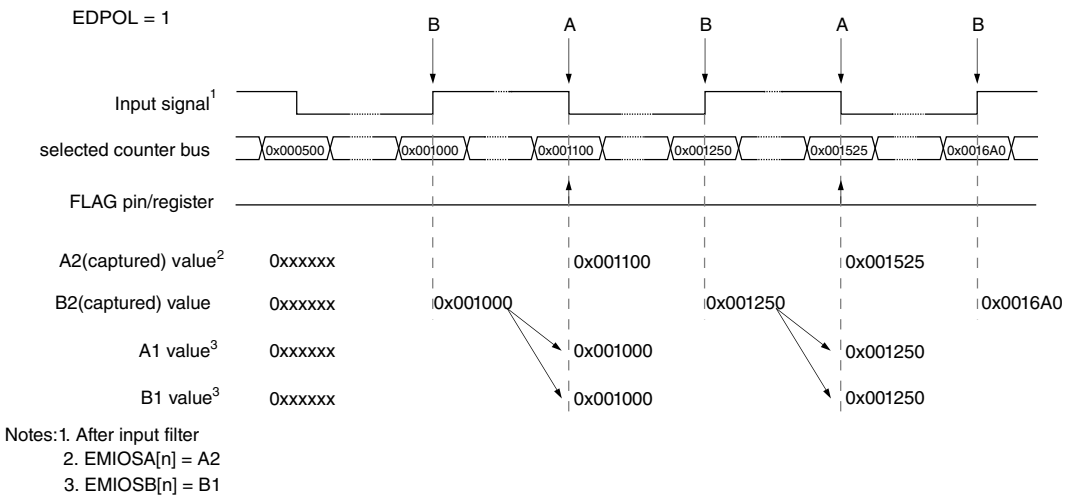


Figure 37-12. Input Pulse Width Measurement example

The following figure shows the A1 and B1 updates when EMIOSA[n] and EMIOSB[n] register reads occur. Note that A1 register has always coherent data related to A2 register. Note also that when EMIOSA[n] read is performed B1 register is loaded with A1 register content. This guarantee that the data in register B1 has always the coherent data related to the last EMIOSA[n] read. The B1 register updates remains locked until EMIOSB[n] read occurs. If EMIOSA[n] read is performed B1 is updated with A1 register content even if B1 update is locked by a previous EMIOSA[n] read operation.

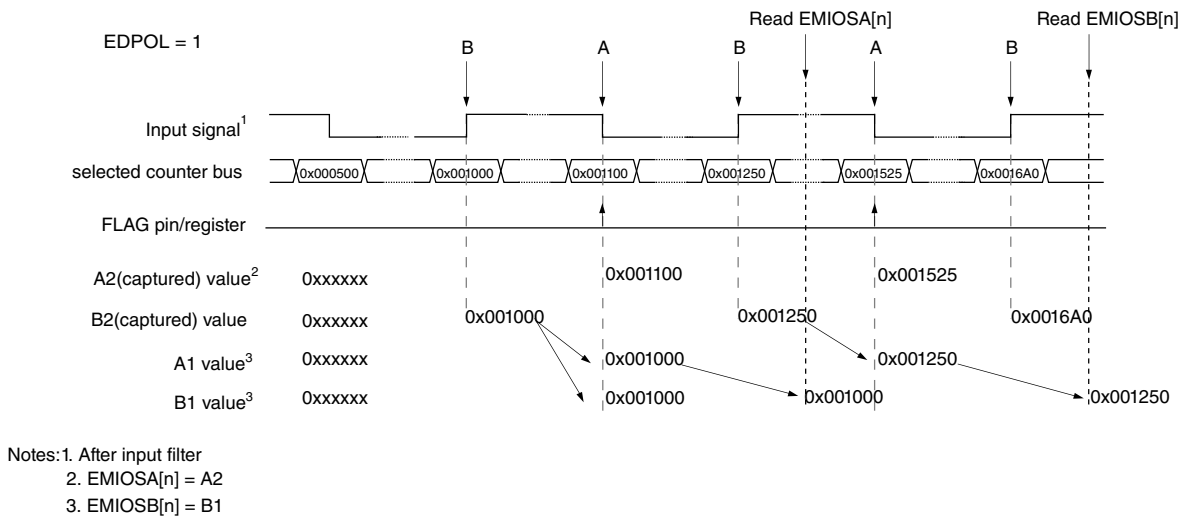


Figure 37-13. B1 and A1 updates at EMIOSA[n] and EMIOSB[n] reads

Reading EMIOS_A_n followed by EMIOS_B_n always provides coherent data. If coherent data is not required for any reason, the sequence of reads should be inverted, therefore EMIOSB[n] should be read prior to EMIOSA[n] register. Note that even in this case B1 register updates will be blocked after EMIOSA[n] read, thus a second EMIOSB[n] is required in order to release B1 register updates.

37.6.1.1.5 Input Period Measurement (IPM) Mode

The IPM mode (MODE[0:6]=0000101) allows the measurement of the period of an input signal by capturing two consecutive rising edges or two consecutive falling edges. Successive input captures are done on consecutive edges of the same polarity. The edge polarity is defined by the EDPOL bit in the EMIOSC[n] register.

Note

The input signal must have at least four system clock cycles period in order to be properly captured by the synchronization logic at the channel input even if the input filter is in by-pass mode.

When the first edge of selected polarity is detected, the selected time base is latched into the registers A2 and B2, and the data previously held in register B2 is transferred to register B1. On this first capture the FLAG line is not set, and the values in registers B1 is meaningless. On the second and subsequent captures, the FLAG line is set and data in register B2 is transferred to register B1.

When the second edge of the same polarity is detected, the counter bus value is latched into registers A2 and B2, the data previously held in register B2 is transferred to data register B1 and to register A1. The FLAG bit is set to indicate the start and end points of a complete period have been captured. This sequence of events is repeated for each subsequent capture. Registers EMIOSA[n] and EMIOSB[n] return the values in register A2 and B1, respectively.

In order to allow coherent data, reading EMIOSA[n] forces A1 content be transferred to B1 register and disables transfers between B2 and B1. These transfers are disabled until the next read of the EMIOSB[n] register. Reading EMIOSB[n] register forces A1 content to be transferred to B1 and re-enables transfers from B2 to B1, to take effect at the next edge capture.

The input pulse period is calculated by subtracting the value in B1 from A2.

The following figure shows how the Unified Channel can be used for input period measurement.

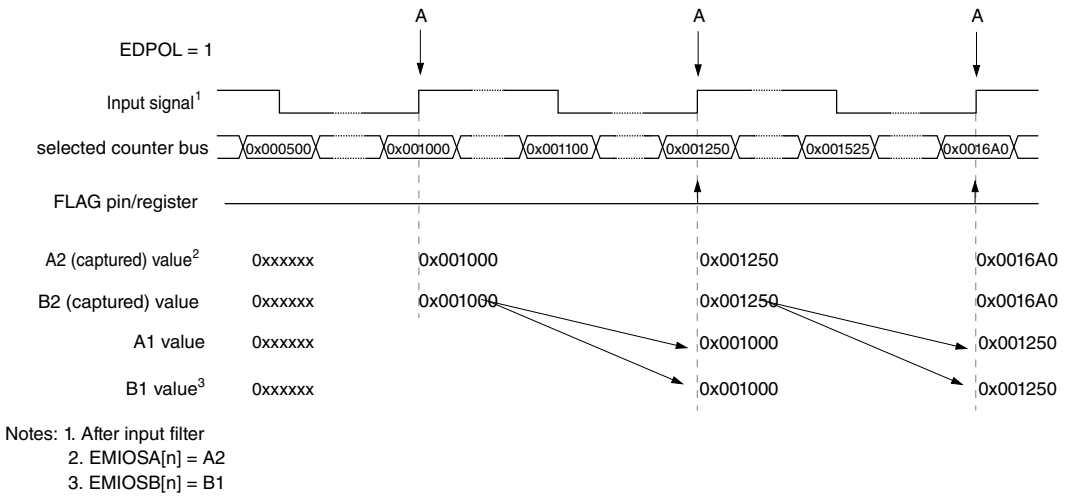


Figure 37-14. Input Period Measurement example

Figure 37-15 describes the A1 and B1 register updates when EMIOSA[n] and EMIOSB[n] read operations are performed. When EMIOSA[n] read occurs the content of A1 is transferred to B1 thus providing coherent data in A2 and B1 registers. Transfers from B2 to B1 are then blocked until EMIOSB[n] is read. After EMIOSB[n] is read, register A1 content is transferred to register B1 and the transfers from B2 to B1 are re-enabled to occur at the transfer edges, which is the leading edge in the following example figure.

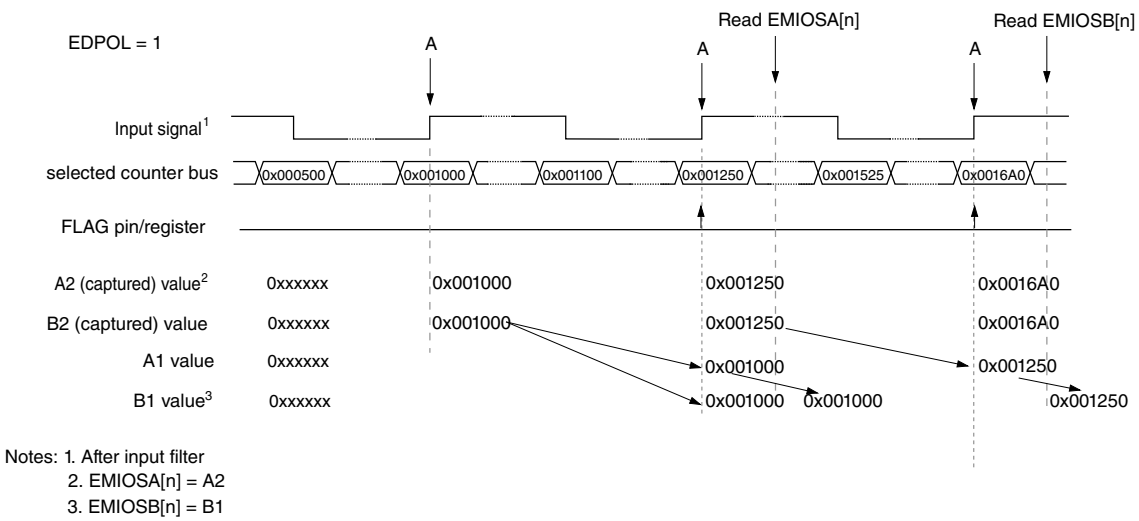


Figure 37-15. A1 and B1 updates at EMIOSA[n] and EMIOSB[n] reads

37.6.1.1.6 Double Action Output Compare (DAOC) Mode

In the DAOC mode, the leading and trailing edges of the variable pulse width output are generated by matches occurring on comparators A and B. There is no restriction concerning the order in which A and B matches occur.

When the DAOC mode is entered, coming out from GPIO mode both comparators are disabled and the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

Data written to A2 and B2 are transferred to A1 and B1, respectively, on the next system clock cycle if OU[n] bit of EMIOSOUDIS register is cleared (see Figure 37-31). The transfer is blocked if OU[n] bit is set. Comparator A is enabled only after the transfer to A1 register occurs and is disabled on the next A match. Comparator B is enabled only after the transfer to B1 register occurs and is disabled on the next B match. Comparators A and B are enabled and disabled independently.

The output flip-flop is set to the value of EDPOL when a match occurs on comparator A and to the complement of EDPOL when a match occurs on comparator B.

MODE[6] controls if the FLAG is set on both matches (MODE[0:6]=0000111) or just on the B match (MODE[0:6]=0000110). FLAG bit assertion depends on comparator enabling.

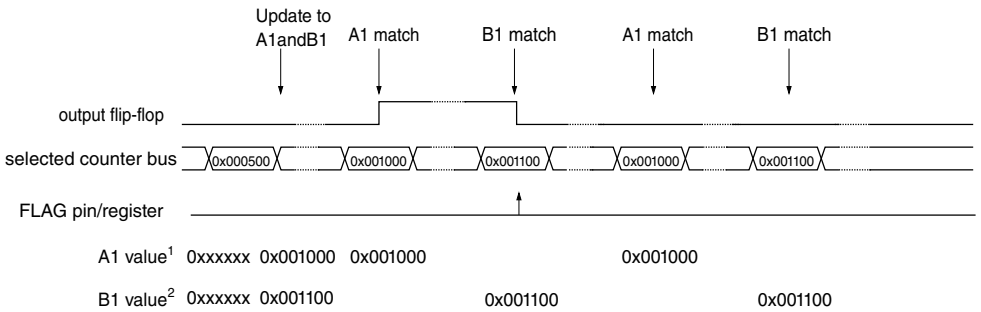
If subsequent enabled output compares occur on registers A1 and B1, pulses will continue to be generated, regardless of the state of the FLAG bit.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a comparison event in comparator A or B, respectively. Note that the FLAG bit is not affected by these forced operations.

Note

If both registers (A1 and B1) are loaded with the same value, the B match prevails concerning the output pin state (output flip-flop is set to the complement of EDPOL), the FLAG bit is set and both comparators are disabled.

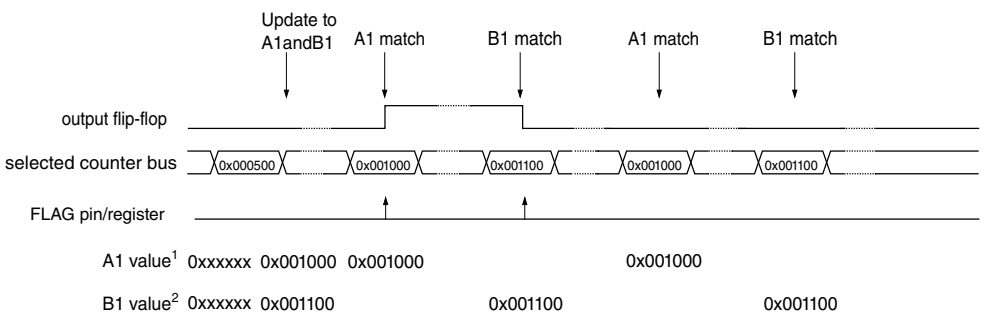
The following two tables show how the Unified Channel can be used to generate a single output pulse with FLAG bit being set on the second match or on both matches, respectively.



Notes:1. EMIOSA[n] = A1 (when reading)
2. EMIOSB[n] = B1 (when reading)

A2 = A1 according to OU[n] bit
B2 = B1 according to OU[n] bit

Figure 37-16. Double Action Output Compare with FLAG set on the second match, MODE[0] bit is 0



Notes:1. EMIOSA[n] = A1 (when reading)
2. EMIOSB[n] = B1 (when reading)

A2 = A1 according to OU[n] bit
B2 = B1 according to OU[n] bit

Figure 37-17. Double Action Output Compare with FLAG set on both matches, MODE[0] bit is 1

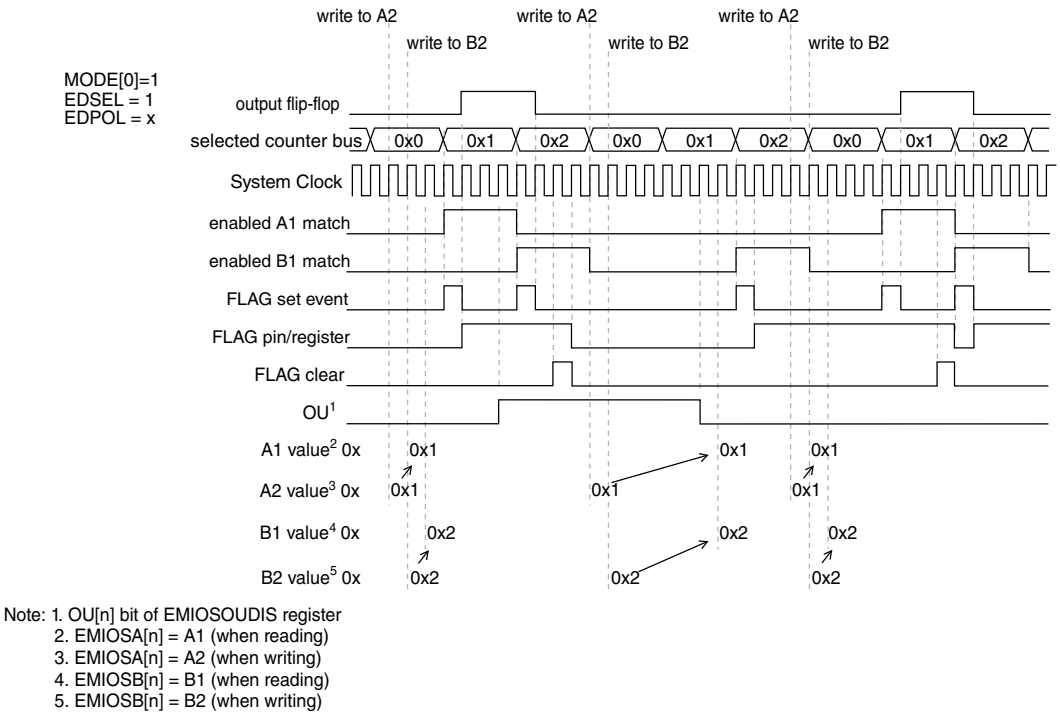


Figure 37-18. DAOC with transfer disabling example

37.6.1.1.7 Modulus Counter (MC) Mode

The MC mode can be used to provide a time base for a counter bus or as a general purpose timer.

MODE[6] bit selects internal or external clock source when cleared or set, respectively. When external clock is selected, the input signal pin is used as the source and the triggering polarity edge is selected by the EDPOL and EDSEL in the EMIOSC[n] register.

The internal counter counts up from the current value until it matches the value in register A1. Register B1 is cleared and is not accessible to the MCU. MODE[4] bit selects up mode or up/down mode, when cleared or set, respectively.

When in up count mode, a match between the internal counter and register A1 sets the FLAG and clears the internal counter. The timing of those events varies according to the MC mode setup as follows:

- Internal counter clearing on match start (MODE[0:6]=001000b)

- External clock is selected if MODE[0] is set. In this case the internal counter clears as soon as the match signal occurs. The channel FLAG is set at the same time the match occurs. Note that by having the internal counter cleared as soon as the match occurs and incremented at the next input event a shorter zero count is generated. See Figure 37-43 and Figure 37-44.
- Internal clock source is selected if MODE[0] is cleared. In this case the counter clears as soon as the match signal occurs. The channel FLAG is set at the same time the match occurs. At the next prescaler tick after the match the internal counter remains at zero and only resumes counting on the following tick. See Figure 37-43 and Figure 37-45.

- Internal counter clearing on match end (MODE[0:6]=001001b)
  - External clock is selected if MODE[0] is set. In this case the internal counter clears when the match signal is asserted and the input event occurs. The channel FLAG is set at the same time the counter is cleared. See Figure 37-43 and Figure 37-46.
  - Internal clock source is selected if MODE[0] is cleared. In this case the internal counter clears when the match signal is asserted and the prescaler tick occurs. The channel FLAG is set at the same time the counter is cleared. See Figure 37-43 and Figure 37-46.

Note

If internal clock source is selected and the prescaler of the internal counter is set to 1 the MC mode behaves the same way even in Clear on Match Start or Clear on Match End sub-modes.

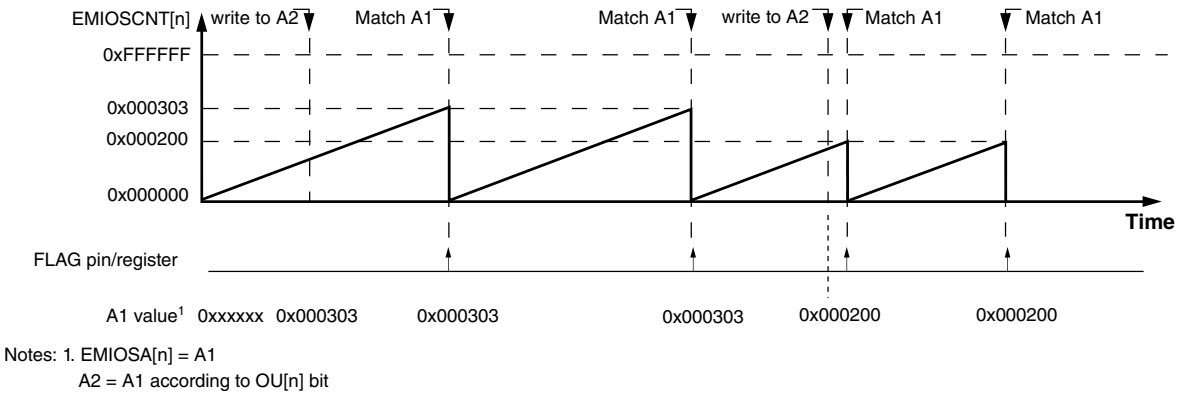
When in up/down count mode (MODE[0:6]=00101bb), a match between the internal counter and register A1 sets the FLAG and changes the counter direction from increment to decrement. A match between register B1 and the internal counter changes the counter direction from decrement to increment and sets the FLAG only if MODE[5] bit is set.

Only values different than 0x0 must be written at A register. Loading 0x0 leads to unpredictable results.

Updates on register A or counter in MC mode may cause loss of match in the current cycle if the transfer occurs near the match. In this case, the counter may rollover and resume operation in the next cycle.

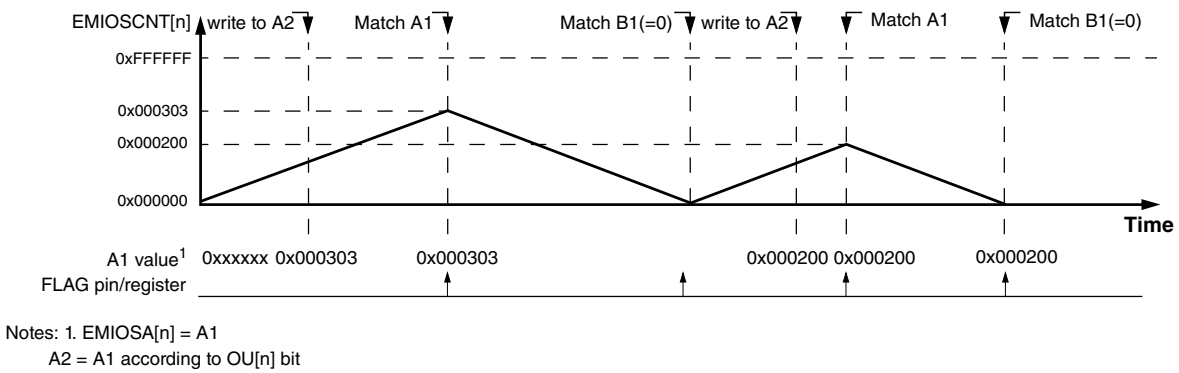
Register B2 has no effect in MC mode. Nevertheless, register B2 can be accessed for reads and writes by addressing EMIOSB.

The following two figures show how the Unified Channel can be used as modulus counter in up mode and up/down mode, respectively.



Notes: 1. EMIOSA[n] = A1
A2 = A1 according to OU[n] bit

Figure 37-19. Modulus Counter up mode example (MODE[4] bit is 0)



Notes: 1. EMIOSA[n] = A1
A2 = A1 according to OU[n] bit

Figure 37-20. Modulus Counter up/down mode example (MODE[4] bit is 1)

37.6.1.1.8 Modulus Counter Buffered (MCB) Mode

The MCB mode provides a time base which can be shared with other channels through the internal counter buses. Register A1 is double buffered thus allowing smooth transitions between cycles when changing A2 register value on the fly. Register A1 is updated at the cycle boundary, which is defined as when the internal counter transitions to 0x1.

The internal counter values operates within a range from 0x1 up to register A1 value. If when entering MCB mode coming out from GPIO mode the internal counter value is not within that range then the A match will not occur causing the channel internal counter to wrap at the maximum counter value which is 0xff_ffff for a 24-bit counter. After the counter wrap occurs it returns to 0x1 and resume normal MCB mode operation. Thus in order to avoid the counter wrap condition make sure its value is within the 0x1 to register A1 value range when the MCB mode is entered.

MODE[6] bit selects internal clock source if cleared or external if set. When external clock is selected the input channel pin is used as the channel clock source. The active edge of this clock is defined by EDPOL and EDSEL bits in the EMIOSC[n] channel register.

When entering in MCB mode, if up counter is selected by MODE[4]=0 (MODE[0:6]=101000b), the internal counter starts counting from its current value to up direction until A1 match occurs. The internal counter is set to 0x1 when its value matches A1 value and a clock tick occurs (either prescaled clock or input pin event).

If up/down counter is selected by setting MODE[4]=1, the counter changes direction at A1 match and counts down until it reaches the value 0x1. After it has reached 0x1 it is set to count in up direction again. Register B1 is used to generate a match in order to set the internal counter in up-count direction if up/down mode is selected. Register B1 cannot be changed while this mode is selected.

Note that differently from the MC mode, the MCB mode counts between 0x1 and A1 register value. Only values greater than 0x1 must be written at register A1. Loading values other than those leads to unpredictable results. The counter cycle period is equal to A1 value in up counter mode. If in up/down counter mode the period is defined by the expression: (2*A1)-2.

Figure 37-21 describes the counter cycle for several A1 values. Register A1 is loaded with A2 register value at the cycle boundary. Thus any value written to A2 register within cycle **n** will be updated to A1 at the next cycle boundary and therefore will be used on cycle **n+1**. The cycle boundary between cycle **n** and cycle **n+1** is defined as when the internal counter transitions from A1 value in cycle **n** to 0x1 in cycle **n+1** . Note that the FLAG is generated at the cycle boundary and has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

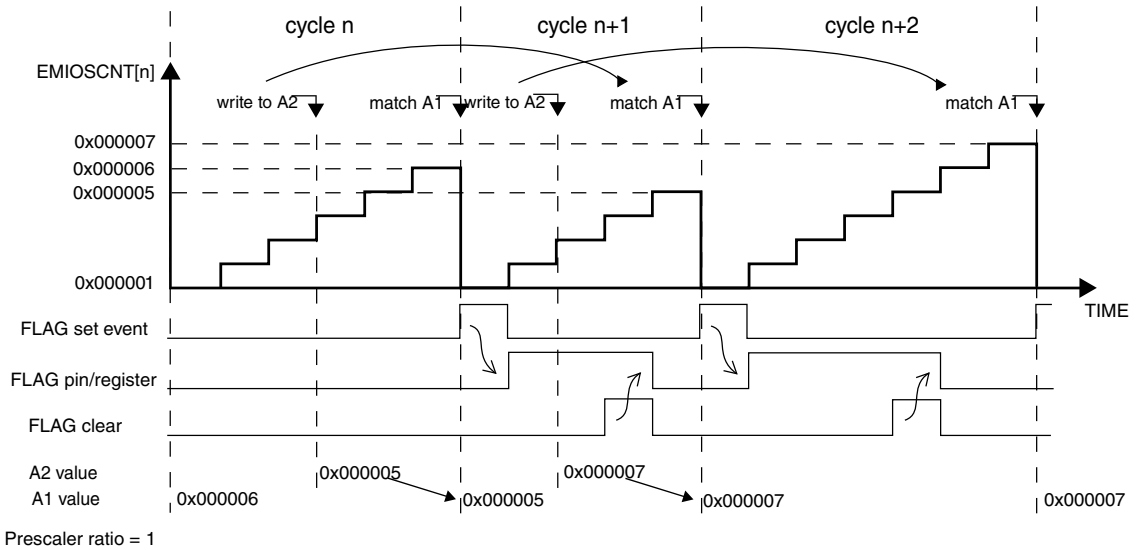


Figure 37-21. Modulus Counter Buffered (MCB) Up Count mode

Figure 37-22 describes the MCB in up/down counter mode (MODE[0:6]=10101bb). Register A1 is updated at the cycle boundary. If A2 is written in cycle **n**, this new value will be used in cycle **n+1** for A1 match. Flags are generated only at A1 match start if MODE[5] is 0. If MODE[5] is set to 1 flags are also generated at the cycle boundary.

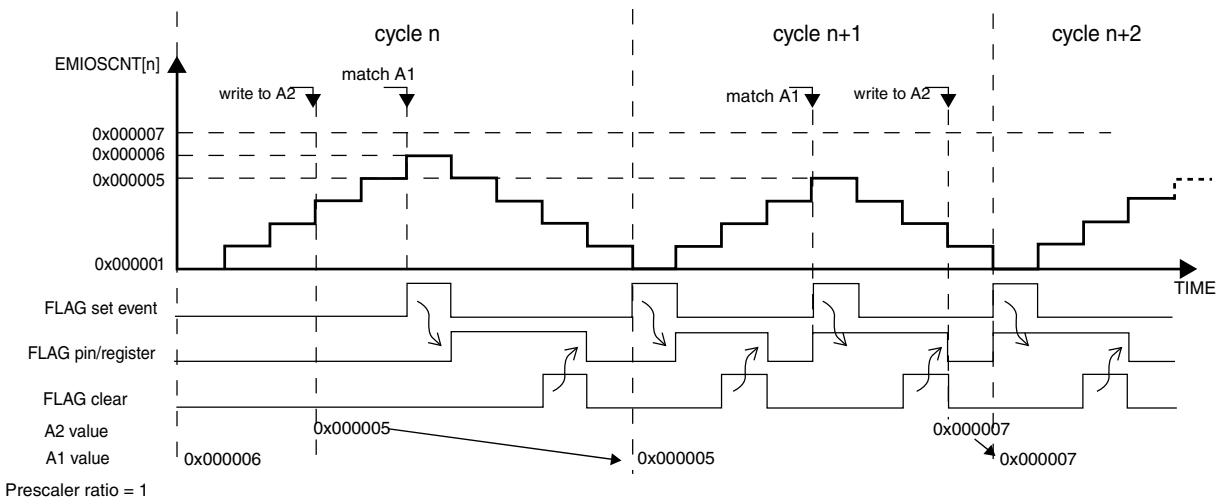


Figure 37-22. Modulus Counter Buffered (MCB) Up/Down Mode

The figure below describes in more detail, the register A1 update process in up counter mode. The A1 load signal is generated at the last system clock period of a counter cycle. Thus, A1 is updated with A2 value at the same time that the counter (EMIOSCNT[n]) is loaded with 0x1. The load signal pulse has the duration of one system clock period. If A2 is written within cycle **n** its value is available at A1 at the first clock of cycle **n+1** and the new value is used for match at cycle **n+1**. The update disable bits OU[n] of EMIOSOUDIS register can be used to control the update of this register, thus allowing to delay the A1 register update for synchronization purposes.

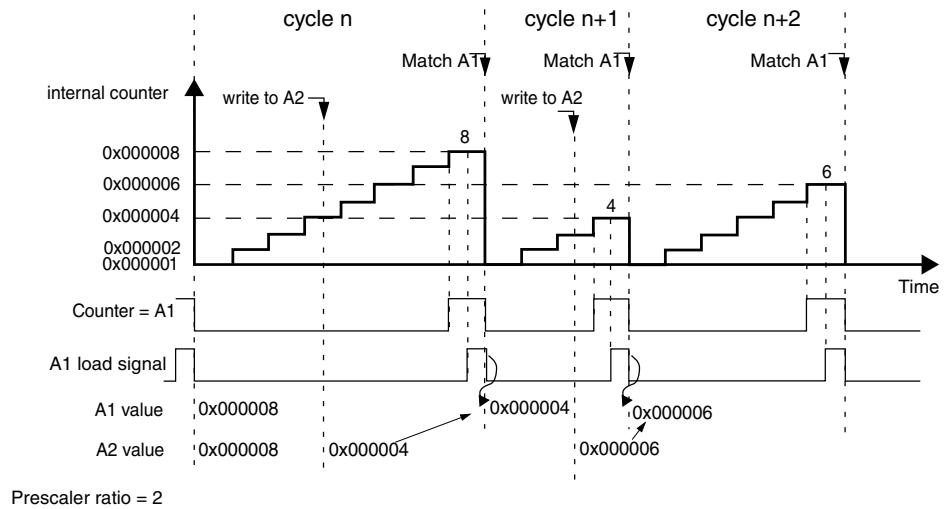


Figure 37-23. MCB Mode Register A1 Update in Up Counter Mode

The figure below describes the register A1 update in up/down counter mode. Note that A2 can be written at any time within cycle **n** in order to be used in cycle **n+1**. Thus A1 receives this new value at the next cycle boundary. Note that the update disable bits OU[n] of EMIOSOUDIS register can be used to disable the update of register A1.

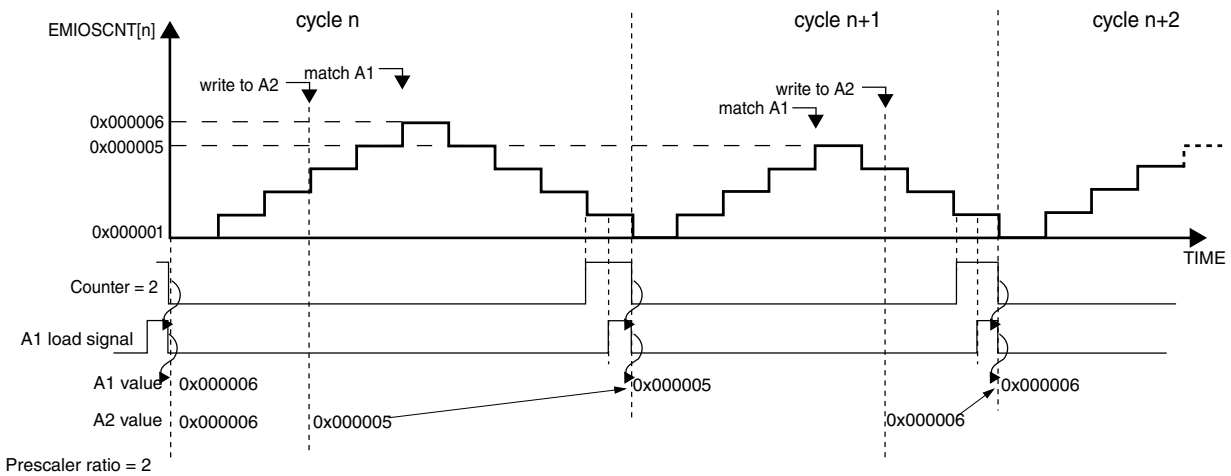


Figure 37-24. MCB Mode Register A1 Update in Up/Down Counter Mode

37.6.1.1.9 Output Pulse Width and Frequency Modulation Buffered (OPWFMB) Mode

This mode (MODE[0:6]=10110b0) provides waveforms with variable duty cycle and frequency. The internal channel counter is automatically selected as the time base when this mode is selected. Register A1 indicates the duty cycle and B1 register the frequency. Both registers A1 and B1 are double buffered to allow smooth signal generation when changing the registers values.. 0% and 100% duty cycles are supported.

At OPWFMB mode entry, the output flip-flop is set to the value of the EDPOL bit in the EMIOSC[n] register.

In order to provide smooth and consistent channel operation this mode differs substantially from the OPWFM mode. The main differences reside in the A1 and B1 registers update, on the delay from the A1 match to the output pin transition and on the range of the internal counter values which starts from 0x1 up to register B1 value.

When entering OPWFMB mode coming out of GPIO mode, the internal counter value is not within that range then the B match will not occur causing the channel internal counter to wrap at the maximum counter value which is 0xff_ffff for a 24-bit counter. After the counter wrap occurs it returns to 0x1 and resume normal OPWFMB mode operation. Thus in order to avoid the counter wrap condition make sure its value is within the 0x1 to B1 register value range when the OPWFMB mode is entered.

When a match on comparator A occurs the output register is set to the value of EDPOL. When a match on comparator B occurs the output register is set to the complement of EDPOL. B1 match also causes the internal counter to transition to 0x1, thus restarting the counter cycle.

Only values greater than 0x1 are allowed to be written to register B1. Loading values other than those leads to unpredictable results.

Figure 37-25 describes the operation of the OPWFMB mode regarding output pin transitions and A1/B1 registers match events. Note that the output pin transition occurs when the A1 or B1 match signal is deasserted which is indicated by the A1 match negedge detection signal. If register A1 is set to 0x4 the output pin transitions 4 counter periods after the cycle had started, plus one system clock cycle. Note that in the example shown in Figure 37-25 the internal counter prescaler has a ratio of two.

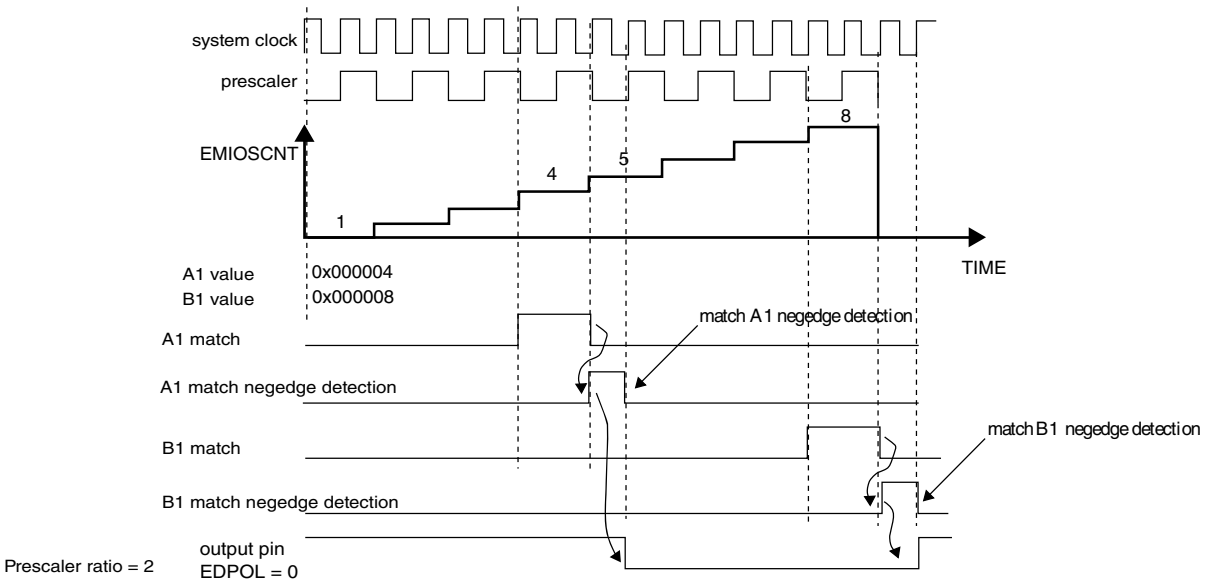


Figure 37-25. OPWFMB A1 and B1 match to Output Register Delay

Figure 37-26 describes the generated output signal if A1 is set to 0x0. Since the counter does not reach zero in this mode, the channel internal logic infers a match as if A1=0x1 with the difference that in this case, the posedge of the match signal is used to trigger the output pin transition instead of the negedge used when A1=0x1. Note that A1 posedge match signal from cycle **n+1** occurs at the same time as B1 negedge match signal from cycle **n** . This allows to use the A1 posedge match to mask the B1 negedge match when they occur at the same time. The result is that no transition occurs on the output flip-flop and a 0% duty cycle is generated.

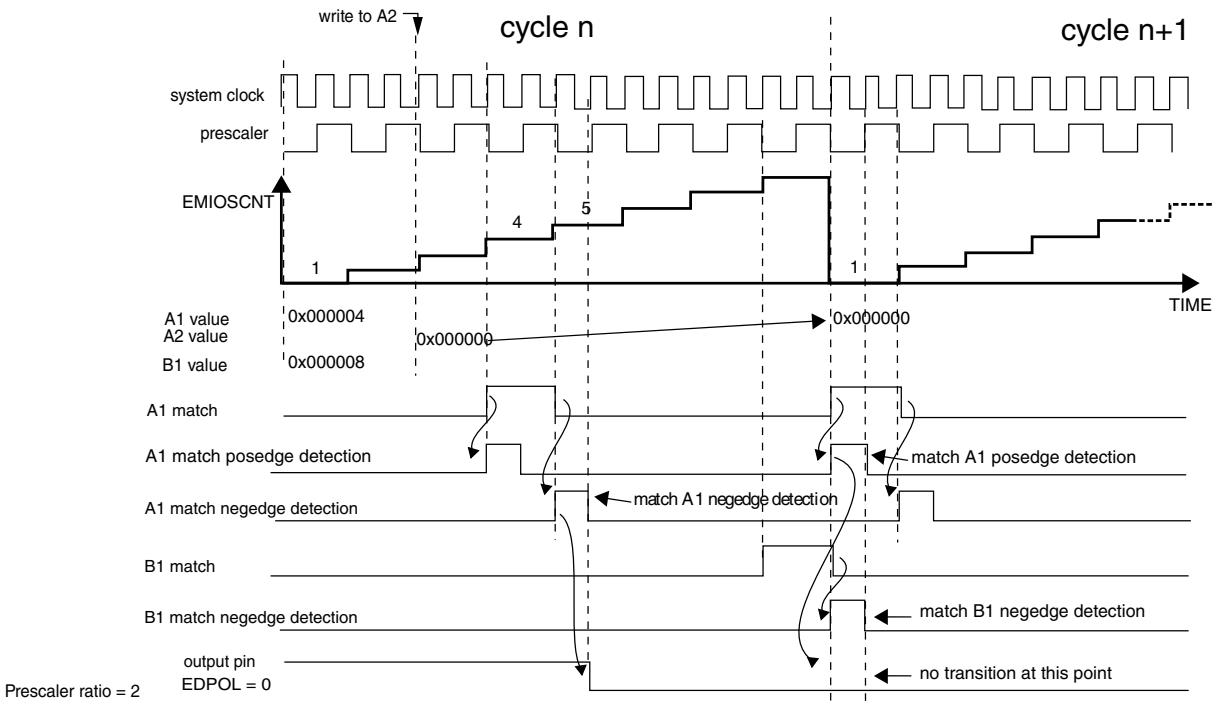


Figure 37-26. OPWFMB Mode with A1 = 0 (0% duty cycle)

Figure 37-27 describes the timing for the A1 and B1 registers load. The A1 and B1 load use the same signal which is generated at the last system clock period of a counter cycle. Thus, A1 and B1 are updated respectively with A2 and B2 values at the same time that the counter (EMIOSCNT[n]) is loaded with 0x1. This event is defined as the cycle boundary. The load signal pulse has the duration of one system clock period. If A2 and B2 are written within cycle **n** their values are available at A1 and B1, respectively, at the first clock of cycle **n+1** and the new values are used for matches at cycle **n+1**. The update disable bits OU[n] of EMIOSOUDIS register can be used to control the update of these registers, thus allowing to delay the A1 and B1 registers update for synchronization purposes.

In Figure 37-27, it is assumed that both the channel and global prescalers are set to 0x1 (each divide ratio is two), meaning that the channel internal counter transitions at every four system clock cycles. FLAGs can be generated only on B1 matches when MODE[5] is cleared, or on both A1 and B1 matches when MODE[5] is set. Since B1 flag occurs at the cycle boundary, this flag can be used to indicate that A2 or B2 data written on cycle **n** were loaded to A1 or B1, respectively, thus generating matches in cycle **n+1**. Note that the FLAG has a synchronous operation, meaning that it is asserted one system clock cycle after the FLAG set event.

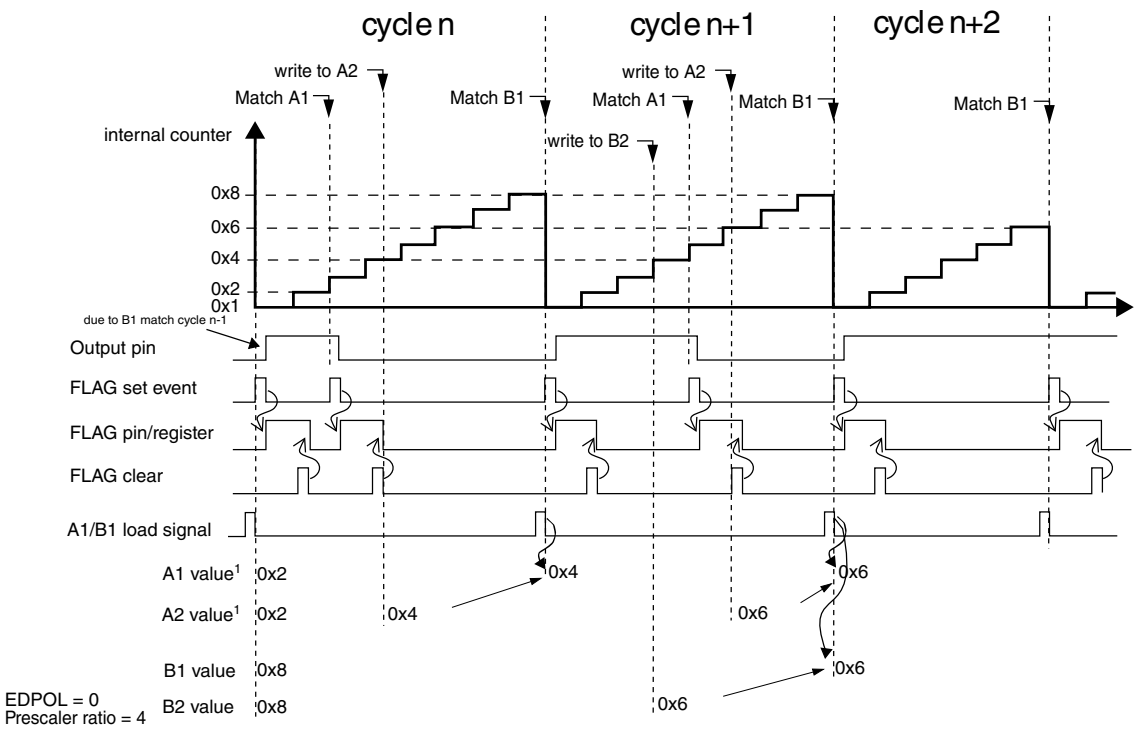


Figure 37-27. OPWFMB A1 and B1 Registers Update and Flags (MODE6] bit is 0b)

Figure 37-28 describes the operation of the Output Disable feature in OPWFMB mode. Differently from the OPWFM mode, the output disable forces the channel output flip-flop to EDPOL bit value. This functionality targets applications that use active high signals and a high to low transition at A1 match. In this case EDPOL should be set to 0. Note that both the channel and global prescalers are set to 0x0 (each divide ratio is one), meaning that the channel internal counter transitions at every system clock cycle.

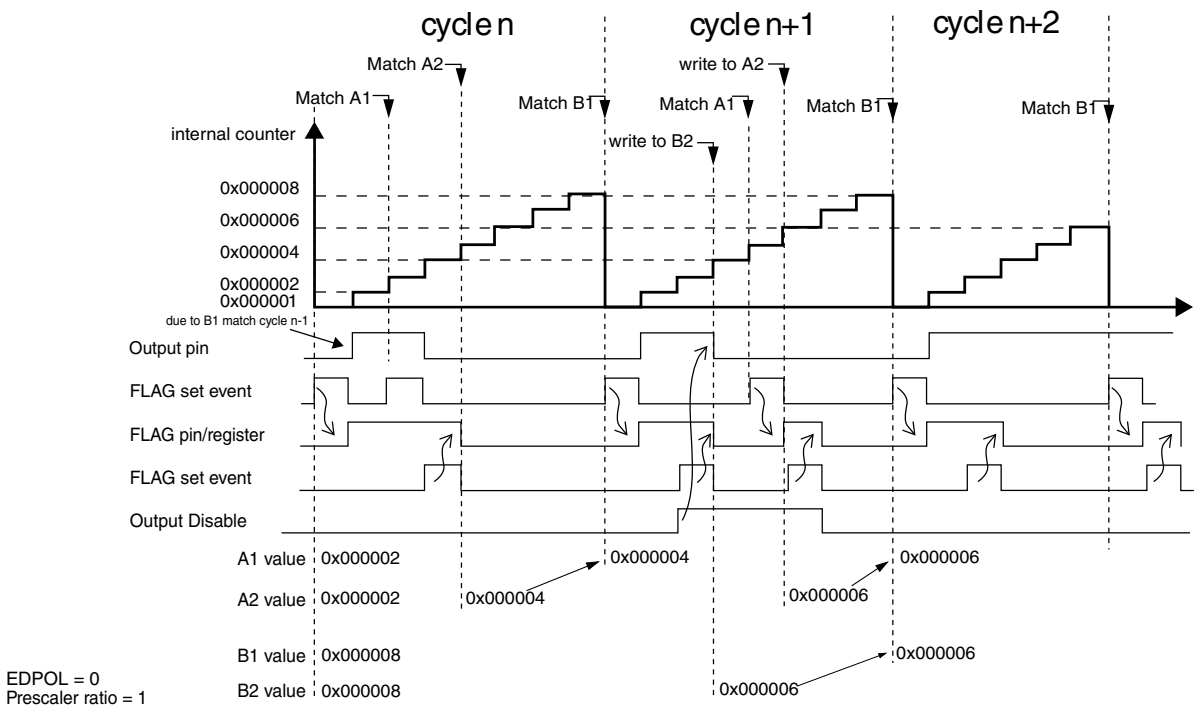


Figure 37-28. OPWFMB Mode with Active Output Disable (MODE[6] bit is 1b)

Note that the output disable has a synchronous operation, meaning that the assertion of the Output Disable input pin causes the channel output flip-flop to transition to EDPOL at the next system clock cycle. If the Output Disable input is deasserted the output pin transition at the following A1 or B1 match.

In Figure 37-28, it is assumed that the Output Disable input is enabled and selected for the Channel. Please, refer to UC Control Register (EMIOSCn) for a detailed description about the ODIS and ODISSL bits, respectively enable and selection of the Output Disable inputs.

The FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on comparators A or B respectively. Similarly to a B1 match FORCMB sets the internal counter to 0x1. The FLAG bit is not set by the FORCMA or FORCMB bits being asserted.

Figure 37-29 describes the generation of 100% and 0% duty cycle signals. It is assumed EDPOL =0 and the resultant prescaler value is 1. Initially, A1=0x8 and B1=0x8. In this case, B1 match has precedence over A1 match, thus the output flip-flop is set to the complement of EDPOL bit. This cycle corresponds to a 100% duty cycle signal. The same output signal can be generated for any A1 value greater or equal to B1.

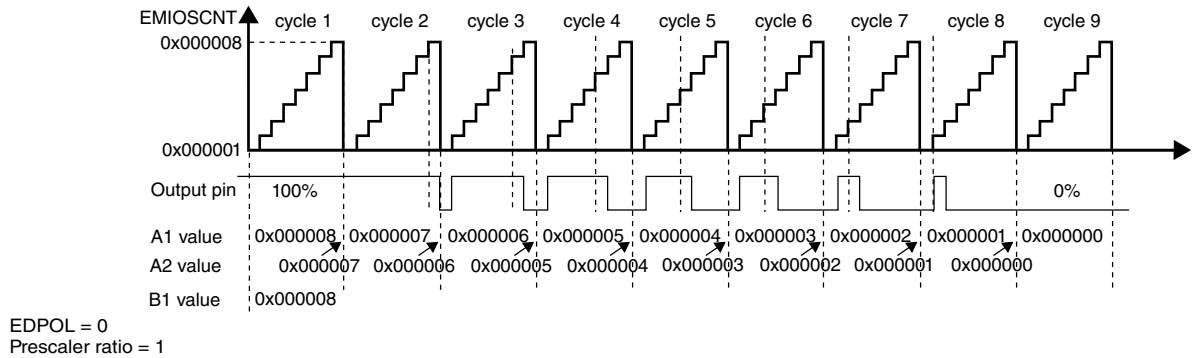


Figure 37-29. OPWFMB Mode from 100% to 0% Duty Cycle

A 0% duty cycle signal is generated if A1=0x0 as shown in Figure 37-29 cycle 9. In this case B1=0x8 match from cycle 8 occurs at the same time as the A1=0x0 match from cycle 9. Please, refer to Figure 37-26 for a description of the A1 and B1 match generation. In this case A1 match has precedence over B1 match and the output signal transitions to EDPOL.

37.6.1.1.10 Center Aligned Output Pulse Width Modulation with Dead Time Insertion Buffered (OPWMCB) Mode

This operation mode generates a center aligned PWM with dead time insertion to the leading (MODE[0:6]=10111b1) or trailing edge (MODE[0:6]=10111b0). A1 and B1 registers are double buffered to allow smooth output signal generation when changing A2 or B2 registers values.

BSL[0:1] bits select the time base. The time base selected for a channel configured to OPWMCB mode should be a channel configured to MCB Up/Down mode, as shown in Figure 37-22 . It is recommended to start the MCB channel time base after the OPWMCB mode is entered in order to avoid missing A matches at the very first duty cycle.

Register A1 contains the ideal duty cycle for the PWM signal and is compared with the selected time base.

Register B1 contains the dead time value and is compared against the internal counter. For a leading edge dead time insertion, the output PWM duty cycle is equal to the difference between register A1 and register B1, and for a trailing edge dead time insertion, the output PWM duty cycle is equal to the sum of register A1 and register B1. Mode[6] bit selects between trailing and leading dead time insertion, respectively.

Note

The internal counter runs in the internal prescaler ratio, while the selected time base may be running in a different prescaler ratio.

When OPWMCB mode is entered, coming out from GPIO mode, the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

The following basic steps summarize proper OPWMCB startup, assuming the channels are initially in GPIO mode:

1. *[global]* Disable Global Prescaler.
2. *[MCB channel]* Disable Channel Prescaler.
3. *[MCB channel]* Write 0x1 at internal counter.
4. *[MCB channel]* Set A register.
5. *[MCB channel]* Set channel to MCB Up mode.
6. *[MCB channel]* Set prescaler ratio.
7. *[MCB channel]* Enable Channel Prescaler.
8. *[OPWMCB channel]* Disable Channel Prescaler.
9. *[OPWMCB channel]* Set A register.
10. *[OPWMCB channel]* Set B register.
11. *[OPWMCB channel]* Select time base input through BSL[1:0] bits.
12. *[OPWMCB channel]* Enter OPWMCB mode.
13. *[OPWMCB channel]* Set prescaler ratio.
14. *[OPWMCB channel]* Enable Channel Prescaler.
15. *[global]* Enable Global Prescaler.

The following figure describes the load of A1 and B1 registers which occurs when the selected counter bus transitions from 0x2 to 0x1 . This event defines the cycle boundary. Note that values written to A2 or B2 within cycle **n** are loaded into A1 or B1 registers, respectively, and used to generate matches in cycle **n+1** .

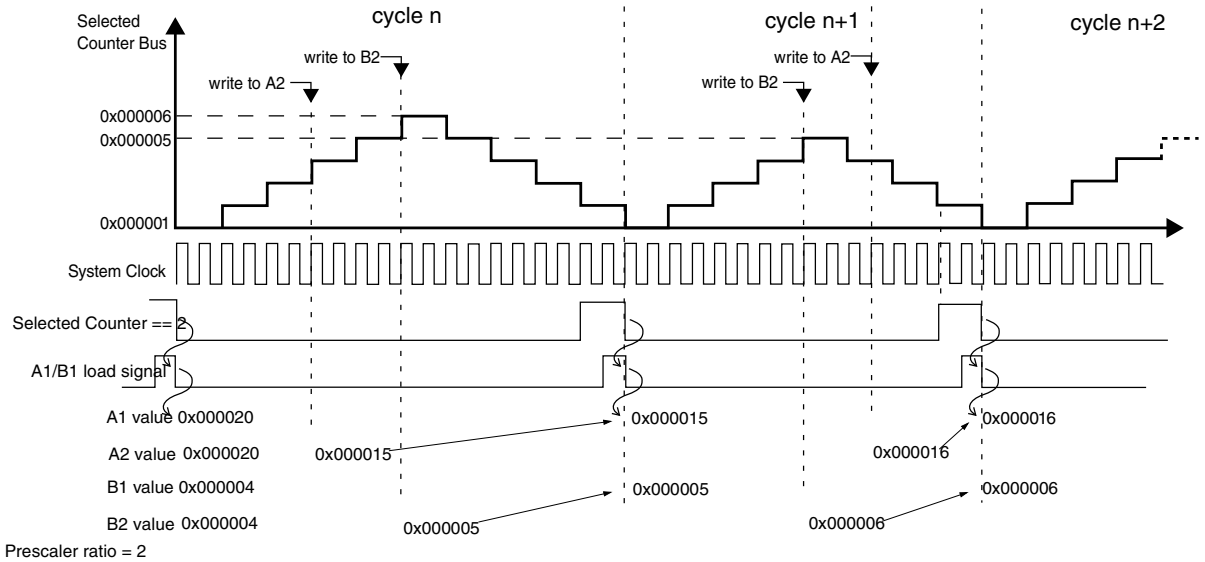


Figure 37-30. OPWMCB A1 and B1 registers load

The OU[n] bit of EMIOSOUDIS register can be used to disable the A1 and B1 updates, thus allowing to synchronize the load on these registers with the load of A1 or B1 registers in others channels. Note that using the update disable bit A1 and B1 registers can be updated at the same counter cycle thus allowing to change both registers at the same time.

In this mode, A1 matches always sets the internal counter to 0x1 . When operating with leading edge dead time insertion the first A1 match sets the internal counter to 0x1. When a match occurs between register B1 and the internal time base, the output flip-flop is set to the value of the EDPOL bit. In the following match between register A1 and the selected time base, the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously. The internal counter should not reach 0x0 as consequence of a rollover. In order to avoid it, the user should not write to EMIOSB register a value greater than twice the difference between external count up limit and EMIOSA value.

The following figure shows two cycles of a Center Aligned PWM signal. Note that both A1 and B1 register values are changing within the same cycle which allows to vary at the same time the duty cycle and dead time values.

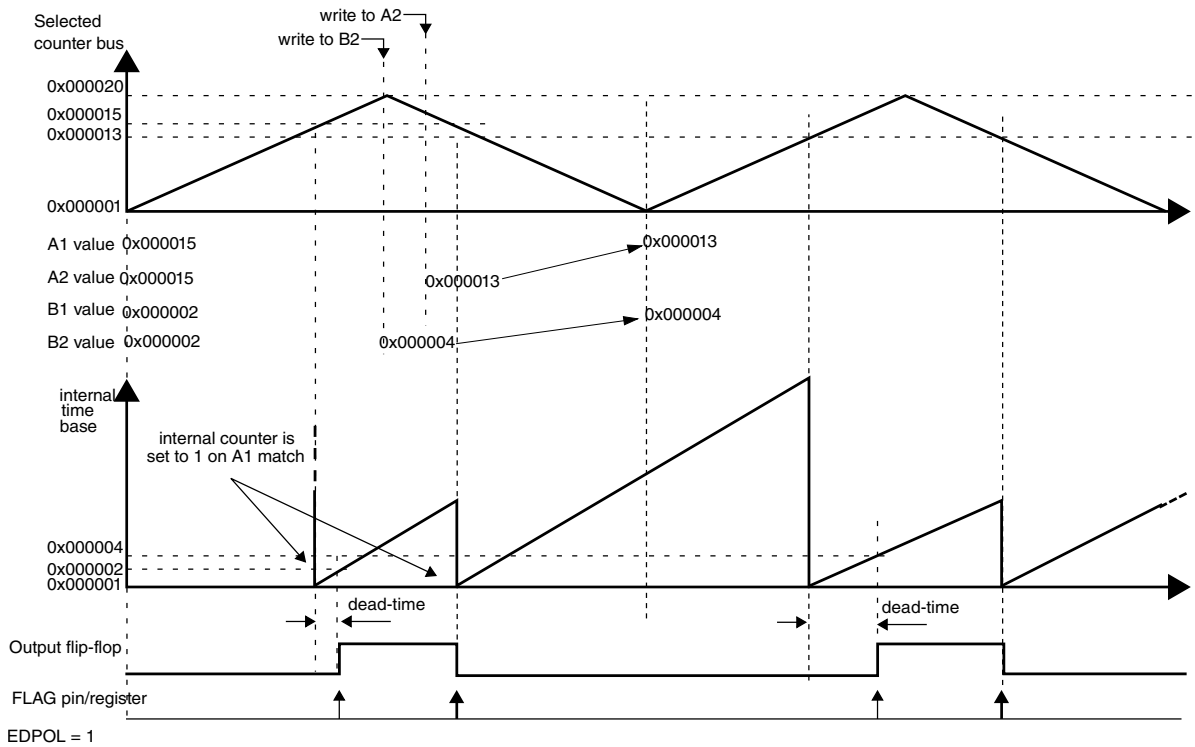


Figure 37-31. OPWMCB with Lead Dead Time Insertion

When operating with trailing edge dead time insertion, the first match between A1 and the selected time base sets the output flip-flop to the value of the EDPOL bit and sets the internal counter to 0x1. In the second match between register A1 and the selected time base, the internal counter is set to 0x1 and B1 matches are enabled. When the match between register B1 and the selected time base occurs the output flip-flop is set to the complement of the EDPOL bit. This sequence repeats continuously.

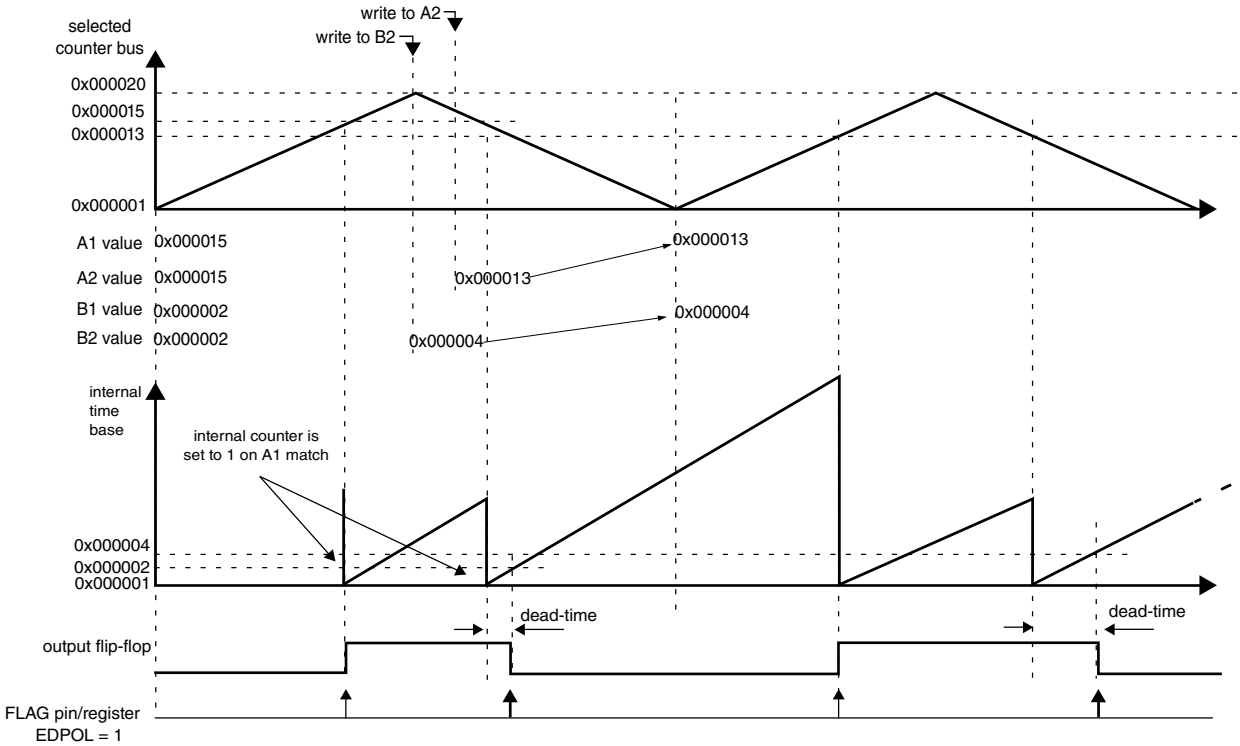


Figure 37-32. OPWMCB with Trail Dead Time Insertion

A FLAG can be generated in the trailing edge of the output PWM signal when MODE[5] is cleared, or in both edges, when MODE[5] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

Note

In OPWMCB mode FORCMA and FORCMB do not have the same behavior as a regular match. Instead they force the output flip-flop to constant value which depends upon the selected dead time insertion mode, lead or trail and the value of the EDPOL bit.

FORCMA has different behaviors depending upon the selected dead time insertion mode, lead or trail. In lead dead time insertion FORCMA force a transition in the output flip-flop to the opposite of EDPOL. In trail dead time insertion the output flip-flop is forced to the value of EDPOL bit.

If FORCMB bit is set, the output flip-flop value depends upon the selected dead time insertion mode. In lead dead time insertion FORCMB forces the output flip-flop to transition to EDPOL bit value. In trail dead time insertion the output flip-flop is forced to the opposite of EDPOL bit value.

Note

FORCMA bit set does not set the internal time-base to 0x1 as a regular A1 match.

The FLAG bit is not set either in case of a FORCMA or FORCMB or even if both forces are issued at the same time.

Note

FORCMA and FORCMB have the same behavior even in Freeze or normal mode regarding the output pin transition.

When *FORCMA* is issued along with *FORCMB* the output flip-flop is set to the opposite of EDPOL bit value. This is equivalent of saying that *FORCMA* has precedence over *FORCMB* when lead dead time insertion is selected and *FORCMB* has precedence over *FORCMA* when trail dead time insertion is selected.

A duty cycle from 0% to 100% can be generated by setting appropriate values to A1 and B1 registers relatively to the period of the external time base. Setting A1=0x1 generates a 100% duty cycle waveform. Assuming EDPOL is set to one and OPWMCB mode with trail dead time insertion, 100% duty cycle signals can be generated if B1 occurs at or after the cycle boundary (external counter = 1). If A1 is greater than the maximum value of the selected counter bus period, then a 0% duty cycle is produced, only if the pin starts the current cycle in the opposite of EDPOL value. In case of 100% duty cycle, the transition from EDPOL to the opposite of EDPOL may be obtained by forcing pin, using FORCMA and/or FORCMB.

Note

If A1 is set to 0x1 at OPWMCB entry the 100% duty cycle may not be obtained in the very first PWM cycle due to the pin condition at mode entry.

Only values different than 0x0 are allowed to be written to A1 register. If 0x0 is loaded to A1, the results are unpredictable.

Note

A special case occurs when A1 is set to (external counter bus period)/2, which is the maximum value of the external counter. In this case the output flip-flop is constantly set to the EDPOL bit value.

The internal channel logic prevents matches from one cycle to propagate to the next cycle. In trail dead time insertion B1 match from cycle **n** could eventually cross the cycle boundary and occur in cycle **n+1.** In this case B1 match is masked out and does not cause the output flip-flop to transition. Therefore matches in cycle **n+1** are not affected by the late B1 matches from cycle **n**.

The following figure shows a 100% duty cycle output signal generated by setting A1=0x4 and B1=0x3. In this case, the trailing edge is positioned at the boundary of cycle **n+1**, which is actually considered to belong to cycle **n+2** and therefore does not cause the output flip-flip to transition.

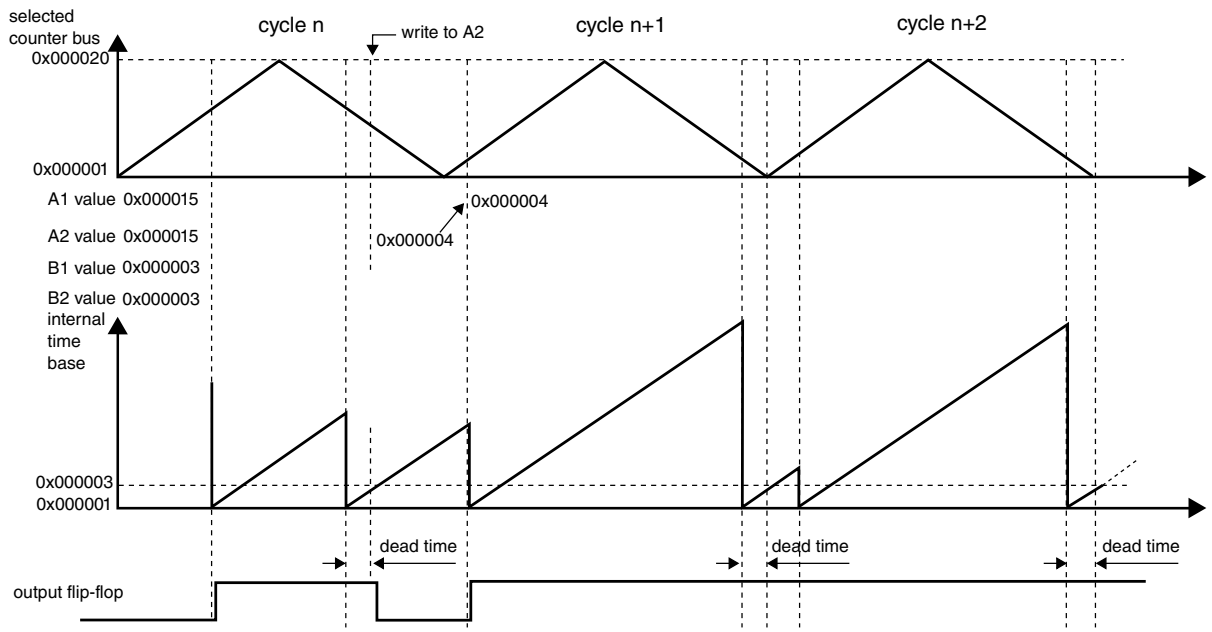


Figure 37-33. OPWMCB with 100% Duty Cycle (A1=4 and B1=3)

The output disable feature, if enabled, causes the output flip-flop to transition to the EDPOL inverted state. This feature allows an application to force the channel output pin to a "safe" state. The internal channel matches continue to occur even in this case, thus generating Flags. As soon as the output disable is deasserted, the channel output pin is again controlled by A1 and B1 matches. Note that this process is synchronous, meaning that the output channel pin transitions only on system clock edges.

It is important to notice that, such as in OPWMB and OPWFMB modes, the match signal used to set or clear the channel output flip-flop is generated on the deassertion of the channel combinational comparator output signal which compares the selected time base with A1 or B1 register values. Please refer to Figure 37-25, which describes the delay from matches to output flip-flop transition in OPWFMB mode. The operation of OPWMCB mode is similar to OPWFMB regarding matches and output pin transition.

37.6.1.1.11 Output Pulse Width Modulation Buffered (OPWMB) Mode

OPWMB mode (MODE[0:6]=11000b0) is used to generate pulses with programmable leading and trailing edge placement. An external counter driven in MCB Up mode must be selected from one of the counter buses. A1 register value defines the first edge and B1

the second edge. The output signal polarity is defined by the EDPOL bit. If EDPOL is zero, a negative edge occurs when A1 matches the selected counter bus and a positive edge occurs when B1 matches the selected counter bus.

The A1 and B1 registers are double buffered and updated from A2 and B2, respectively, at the cycle boundary. The load operation is similar to the OPWFMB mode. Please refer to Figure 37-27 for more information about A1 and B1 registers update.

FLAG can be generated at B1 matches, when MODE[5] is cleared, or in both A1 and B1 matches, when MODE[5] is set. If subsequent matches occur on comparators A and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A1 or B1 respectively. The FLAG bit is not set by the FORCMA and FORCMB operations.

At OPWMB mode entry the output flip-flop is set to the value of the EDPOL bit in the EMIOSC[n] register.

Some rules applicable to the OPWMB mode are:

- B1 matches have precedence over A1 matches if they occur at the same time within the same counter cycle
- A1=0 match from cycle **n** has precedence over B1 match from cycle **n-1**
- A1 matches are masked out if they occur after B1 match within the same cycle
- Any value written to A2 or B2 on cycle **n** is loaded to A1 and B1 registers at the following cycle boundary (assuming OU[n] bit of EMIOSOUDIS register is not asserted). Thus the new values will be used for A1 and B1 matches in cycle **n+1**

The following figure describes the operation of the OPWMB mode regarding A1 and B1 matches and the transition of the channel output pin. In this example EDPOL is set to zero.

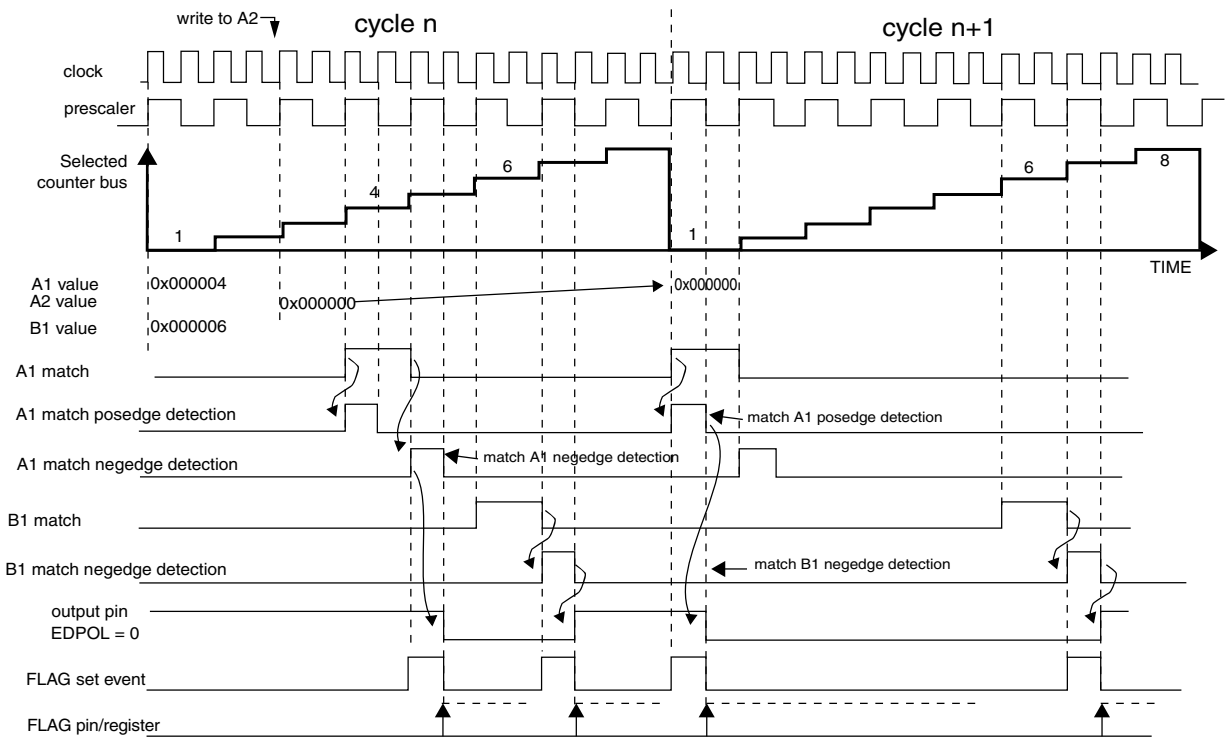


Figure 37-34. OPWMB Mode Matches and Flags

Note that the output pin transitions are based on the negedges of the A1 and B1 match signals. The figure above shows in cycle **n+1** the value of A1 register being set to zero. In this case the match posedge is used instead of the negedge to transition the output flip-flop.

The figure below describes the channel operation for 0% duty cycle. Note that the A1 match posedge signal occurs at the same time as the B1=0x8 negedge signal. In this case A1 match has precedence over B1 match, causing the output pin to remain at EDPOL bit value, thus generating a 0% duty cycle signal.

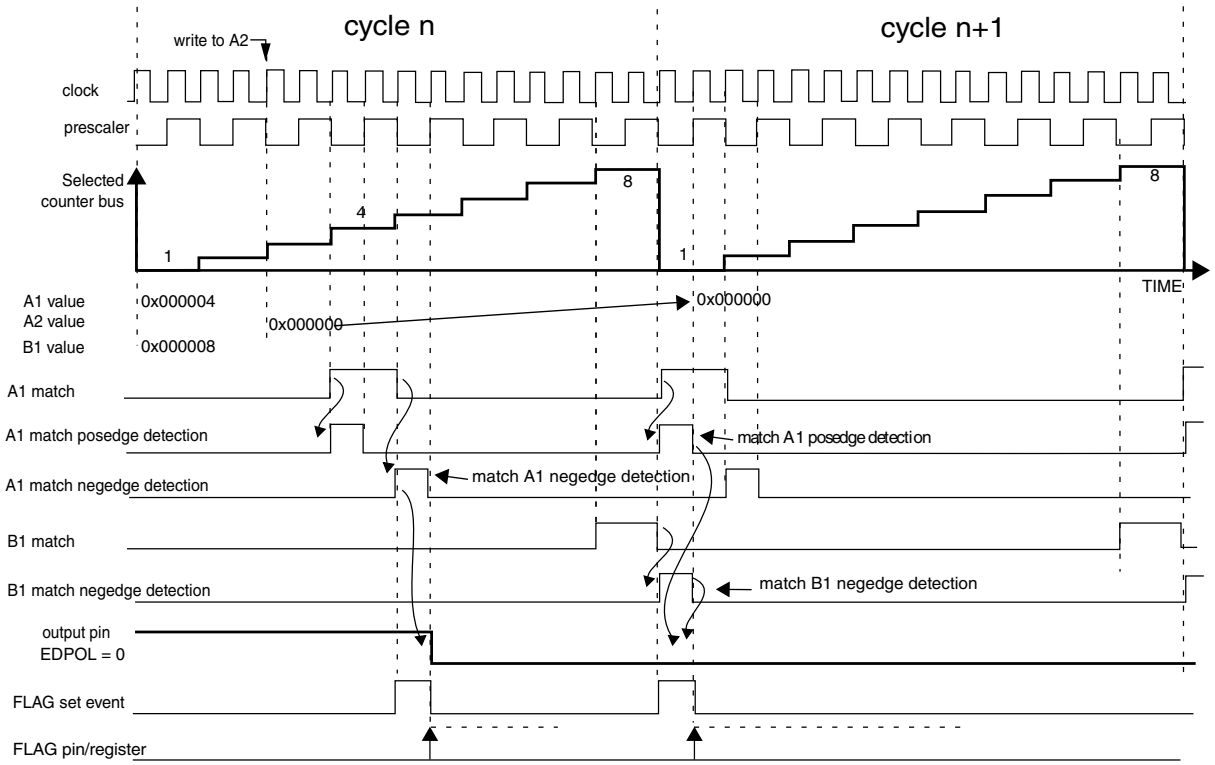


Figure 37-35. OPWMB Mode with 0% Duty Cycle

Figure 37-36 describes the operation of the OPWMB mode with the Output Disable signal being asserted. The Output Disable forces a transition in the output pin to the EDPOL bit value. After deasserted, the output disable allows the output pin to transition at the following A1 or B1 match. Note that the Output Disable does not modify the Flag bit behavior. Note that there is one system clock delay between the assertion of the output disable signal and the transition of the output pin to EDPOL.

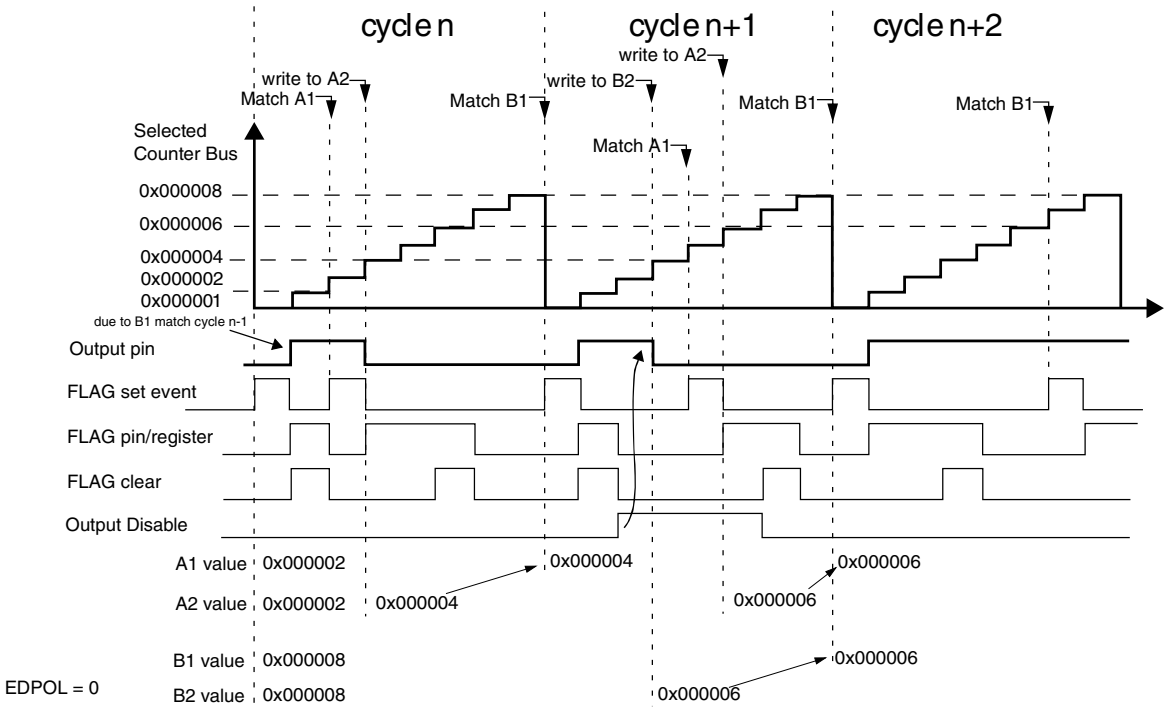


Figure 37-36. OPWMB Mode with Active Output Disable (MODE[6] bit is 1b)

Figure 37-37 shows a waveform changing from 100% to 0% duty cycle. EDPOL in this case is zero. In this example, B1 is programmed to the same value as the period of the external selected time base.

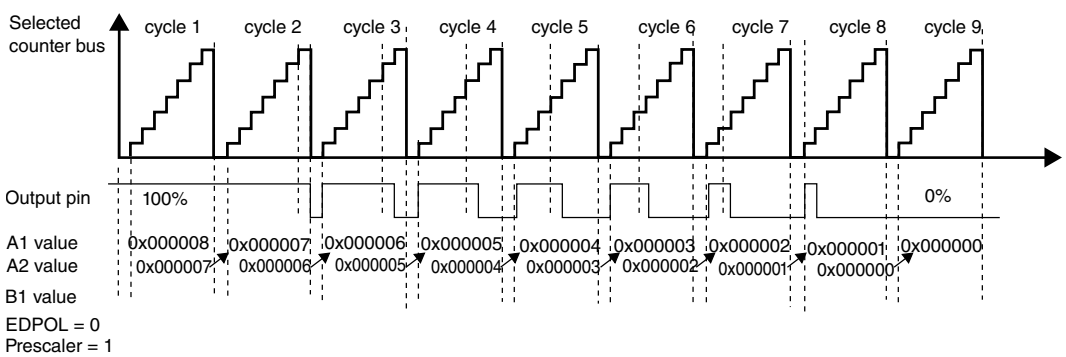


Figure 37-37. OPWMB Mode from 100% to 0% Duty Cycle

In Figure 37-37, if B1 is set to a value lower than 0x8, it is not possible to achieve 0% duty cycle by only changing A1 register value. Since B1 matches have precedence over A1 matches the output pin transitions to the opposite of EDPOL bit at B1 match. Note also that if B1 is set to 0x9, for instance, B1 match does not occur, thus a 0% duty cycle signal is generated.

37.6.1.1.12 Output Pulse Width Modulation with Trigger (OPWMT) Mode

OPWMT mode (MODE[0:6]=0100110) is intended to support the generation of Pulse Width Modulation signals where the period is not modified while the signal is being output, but where the duty cycle will be varied and must not create glitches. The mode is intended to be used in conjunction with other channels executing in the same mode and sharing a common timebase. It will support each channel with a fixed PWM leading edge position with respect to the other channels and the ability to generate a trigger signal at any point in the period that can be output from the module to initiate activity in other parts of the SoC such as starting ADC conversions.

An external counter driven in either MC Up or MCB Up mode must be selected from one of the counter buses.

Register A1 defines the leading edge of the PWM output pulse and as such the beginning of the PWM's period. This makes it possible to insure that the leading edge of multiple channels in OPWMT mode can occur at a specific time with respect to the other channels when using a shared timebase. This can allow the introduction of a fixed offset for each channel which can be particularly useful in the generation of lighting PWM control signals where it is desirable that edges are not coincident with each other to help eliminate noise generation. The value of register A1 represents the shift of the PWM channel with respect to the selected timebase. A1 can be configured with any value within the range of the selected time base. Note that registers loaded with 0x0 will not produce matches if the timebase is driven by a channel in MCB mode.

A1 is not buffered as the shift of a PWM channel must not be modified while the PWM signal is being generated. In case A1 is modified it is immediately updated and one PWM pulse could be lost.

EMIOSB[n] address gives access to B2 register for write and B1 register for read. Register B1 defines the trailing edge of the PWM output pulse and as such the duty cycle of the PWM signal. To synchronize B1 update with the PWM signal and so ensure a correct output pulse generation the transfer from B2 to B1 is done at every match of register A1. This behavior is the same as the OPWM mode with next period update.

EMIOSOUDIS register affects transfers between B2 and B1 only.

In order to account for the shift in the leading edge of the waveform defined by register A1 it will be necessary that the trailing edge, held in register B1, can roll over into the next period. This means that a match against the B1 register should not have to be qualified by a match in the A1 register. The impact of this would mean that incorrectly setting register B1 to a value less that register A1 will result in the output being held over a cycle boundary until the B1 value is encountered.

This mode provides a buffered update of the trailing edge by updating register B1 with register B2 contents only at a match of register A1.

The value loaded in register A1 is compared with the value on the selected time base. When a match on comparator A1 occurs, the output flip-flop is set to the value of the EDPOL bit. When a match occurs on comparator B, the output flip-flop is set to the complement of the EDPOL bit.

Note that the output pin and flag transitions are based on the posedges of the A1, B1 and A2 match signals. Please refer to Figure 37-34 in Output Pulse Width Modulation Buffered (OPWMB) Mode for details on match posedge.

Register A2 defines the generation of a trigger event within the PWM period and A2 should be configured with any value within the range of the selected time base, otherwise no trigger will be generated. A match on the comparator will generate the FLAG signal but it has no effect on the PWM output signal generation. The typical setup to obtain a trigger with FLAG is enabling DMA and driving the channel's ipd_done input high.

A2 is not buffered and therefore its update is immediate. If the channel is running when a change is made this could cause either the loss of one trigger event or the generation of two trigger events within the same period. Register A2 can be accessed by reading or writing the eMIOS UC Alternate A Register (eMIOS_ALTA) at UC[n] base address +0x14.

The FLAG signal is set only at match on the comparator with A2. A match on the comparator with A1 or B1 or B2 has no effect on FLAG.

At any time, the FORCMA and FORCMB bits allow the software to force the output flip-flop to the level corresponding to a match on A or B respectively. Any FORCMA and/or FORCMB has priority over any simultaneous match regarding to output pin transitions. Note that the load of B2 content on B1 register at an A match is not inhibited due to a simultaneous FORCMA/FORCMB assertion. If both FORCMA and FORCMB are asserted simultaneously the output pin goes to the opposite of EDPOL value such as if A1 and B1 registers had the same value. FORCMA assertion causes the transfer from register B2 to B1 such as a regular A match, regardless of FORCMB assertion.

If subsequent matches occur on comparators A1 and B, the PWM pulses continue to be generated, regardless of the state of the FLAG bit.

At OPWMT mode entry the output flip-flop is set to the complement of the EDPOL bit in the EMIOSC[n] register.

In order to achieve 0% duty cycle both registers A1 and B must be set to the same value. When a simultaneous match on comparators A and B occur, the output flip-flop is set at every period to the complement value of EDPOL.

In order to achieve 100% duty cycle the register B1 must be set to a value greater than maximum value of the selected time base. As a consequence if 100% duty cycle must be implemented the maximum counter value for the time base is 0xFFFFFE for a 24-bit counter and 0xFFFE for a 16-bit counter, respectively. When a match on comparator A1 occurs the output flip-flop is set at every period to the value of EDPOL bit. The transfer from register B2 to B1 is still triggered by the match at comparator A.

As with other eMIOS mode, the OPWMT mode will implement the Output Disable function. Setting the ODIS bit in the eMIOS UC Control Register (EMIOSC[n]) will enable the Output Disable function. If the selected Output Disable input signal is asserted for the channel, the output pin will go to the inverse of the EDPOL. The channel will continue to operate normally, although the output will be fixed. When the Output Disable input signal is negated, the output pin will return to operate as normal.

Figure 37-38 shows the Unified Channel running in OPWMT mode with Trigger Event Generation and duty cycle update on next period update.

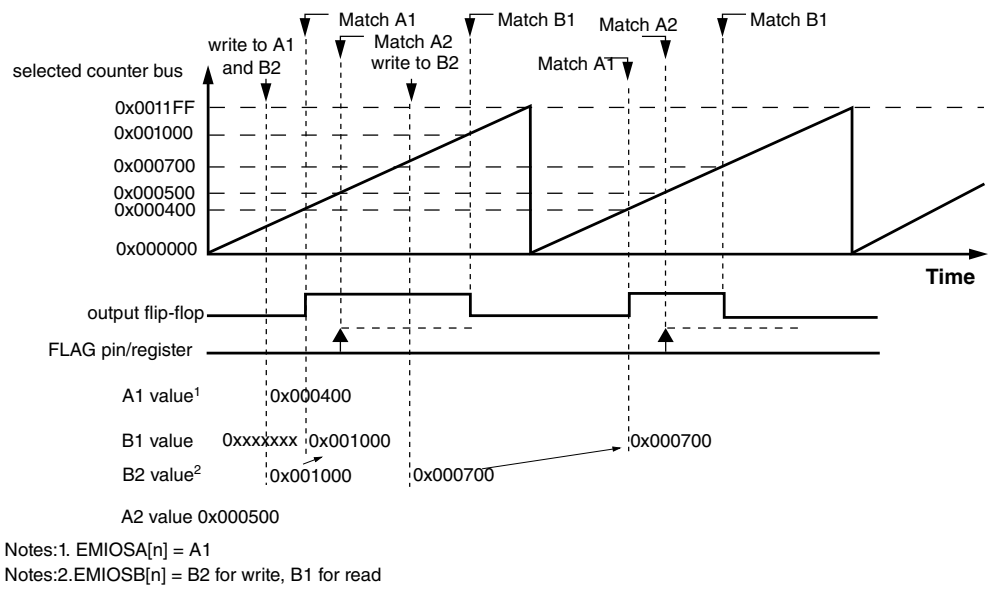


Figure 37-38. OPWMT example

Figure 37-39 shows the Unified Channel running in OPWMT mode with Trigger Event Generation and 0% duty.

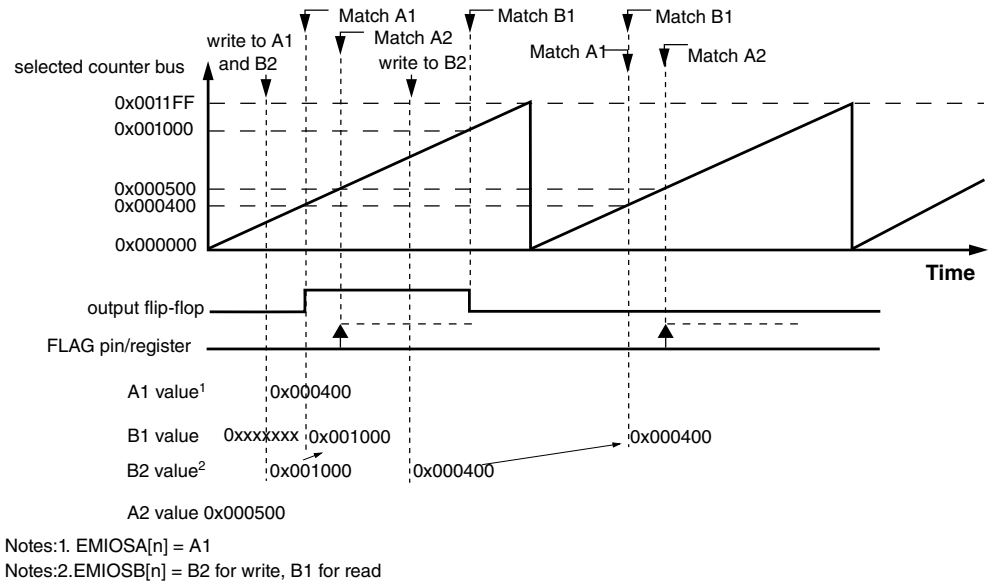


Figure 37-39. OPWMT with 0% Duty Cycle

Figure 37-40 shows the Unified Channel running in OPWMT mode with Trigger Event Generation and 100% duty cycle.

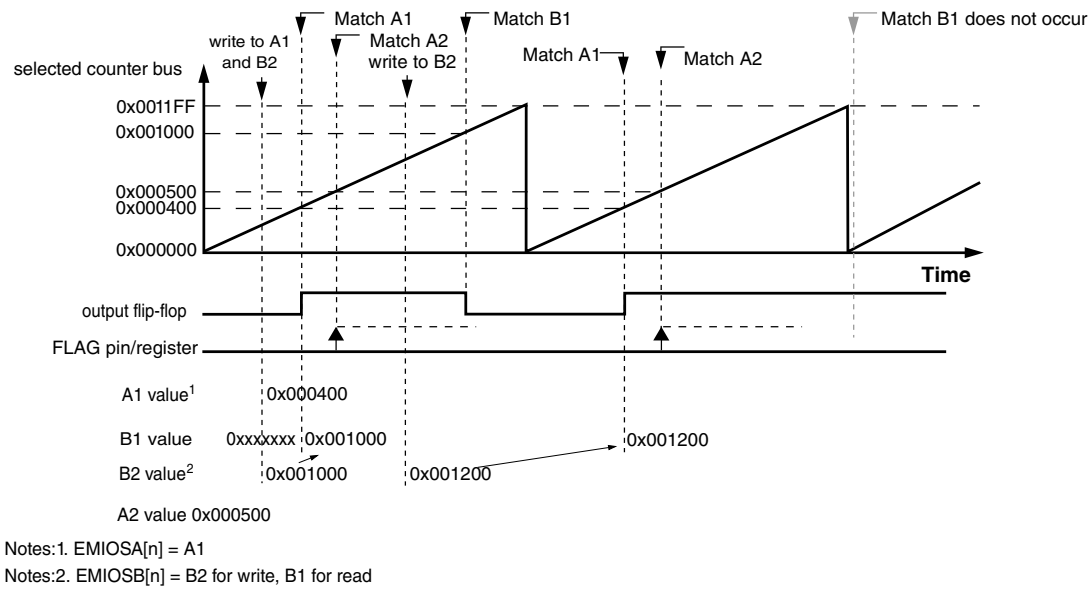


Figure 37-40. OPWMT with 100% Duty Cycle

37.6.1.2 Input Programmable Filter (IPF)

The IPF ensures that only valid input pin transitions are received by the Unified Channel edge detector. A block diagram of the IPF is shown in Figure 37-41.

The IPF is a 5-bit programmable up counter that is incremented by the selected clock source, according to bits IF[0:3] in EMIOSC[n] register.

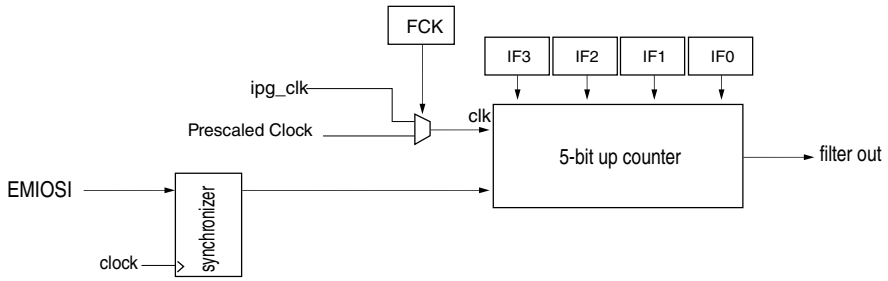


Figure 37-41. Input Programmable Filter Submodule Diagram

The input signal is synchronized by system clock . When a state change occurs in this signal, the 5-bit counter starts counting up. As long as the new state is stable on the pin, the counter remains incrementing. If a counter overflow occurs, the new pin value is validated. In this case, it is transmitted as a pulse edge to the edge detector. If the opposite edge appears on the pin before validation (overflow), the counter is reset. At the next pin transition, the counter starts counting again. Any pulse that is shorter than a full range of the masked counter is regarded as a glitch and it is not passed on to the edge detector. A timing diagram of the input filter is shown in Figure 37-42.

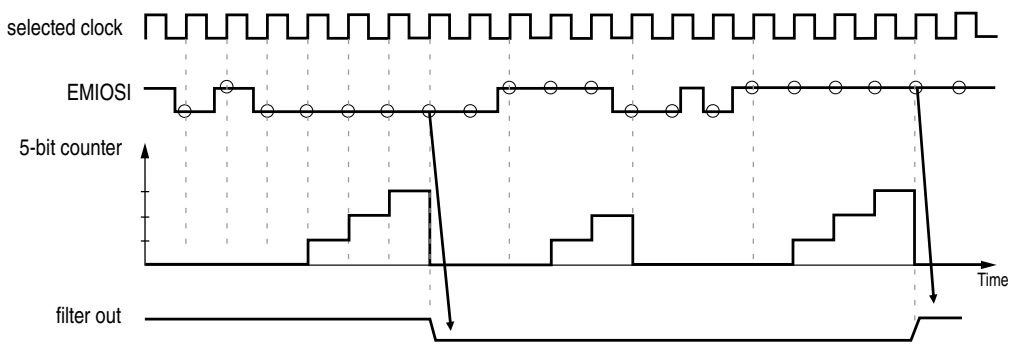


Figure 37-42. Input Programmable Filter Example (IF[0:3] is 0010b)

The filter is not disabled during either freeze state or negated GTBE input

37.6.1.3 Clock Prescaler (CP)

The CP divides the GCP output signal to generate a clock enable for the internal counter of the Unified Channels. The GCP output signal is prescaled by the value defined in the UCPRE[0:1] field description in the EMIOSC[n] register. The prescaler is enabled by setting the UCPREN bit in the EMIOSC[n] and can be stopped at any time by clearing this bit, thereby stopping the internal counter in the Unified Channel.

In order to ensure safe working and avoid glitches the following steps must be performed whenever any update in the prescaling rate is desired:

1. Write 0 at both GPREN bit in EMIOSMCR register and UCPREN bit in EMIOSC[n] register, thus disabling prescalers;
2. Write the desired value for prescaling rate at UCPRE[0:1] bits in EMIOSC[n] register;
3. Enable channel prescaler by writing 1 at UCPREN bit in EMIOSC[n] register;
4. Enable global prescaler by writing 1 at GPREN bit in EMIOSMCR register.

The prescaler is not disabled during either freeze state or negated GTBE input.

37.6.1.4 Effect of Freeze on the Unified Channel

When in debug mode, FRZ bit in the EMIOSMCR register and the FREN bit in the EMIOSC[n] are both set, the internal counter and Unified Channel capture and compare functions are halted. The UC is frozen in its current state.

During freeze, all registers are accessible. When the Unified Channel is operating in an output mode, the force match functions remain available, allowing the software to force the output to the desired level.

Note that for input modes, any input events that may occur while the channel is frozen are ignored.

When exiting debug mode or freeze enable bit is cleared (FRZ in the EMIOSMCR or FREN in the EMIOSC[n] register) the channel actions resume, but may be inconsistent until channel enters GPIO mode again.

37.6.2 IP Bus Interface Unit (BIU)

The BIU provides the interface between the Internal Interface Bus (IIB) and the Peripheral Bus, allowing communication among all submodules and this IP interface.

The BIU allows 8, 16 and 32 bits access. They are performed over a 32-bit data bus in a single cycle clock.

37.6.2.1 Effect of Freeze on the BIU

When the FRZ bit in the EMIOSMCR register is set and the module is in debug mode, the operation of BIU is not affected.

37.6.3 Global Clock Prescaler Submodule (GCP)

The GCP divides the system clock to generate a clock for the CPs of the channels. The main clock signal is prescaled by the value defined in the Global Prescaler clock divider table within the eMIOS Module Configuration Register (EMIOSMCR) description according to the GPRE[0:7] bits in EMIOSMCR register. The global prescaler is enabled by setting the GPREN bit in the EMIOSMCR register and can be stopped at any time by clearing this bit, thereby stopping the internal counters in all the channels.

In order to ensure safe working and avoid glitches the following steps must be performed whenever any update in the prescaling rate is desired:

1. Write 0 at GPREN bit in EMIOSMCR register, thus disabling global prescaler;
2. Write the desired value for prescaling rate at GPRE[0:7] bits in EMIOSMCR register;
3. Enable global prescaler by writing 1 at GPREN bit in EMIOSMCR register.

The prescaler is not disabled during either freeze state or negated GTBE input.

37.6.3.1 Effect of Freeze on the GCP

When the FRZ bit in the EMIOSMCR register is set and the module is in debug mode, the operation of GCP submodule is not affected, i.e., there is no freeze function in this submodule.

37.7 Initialization/Application Information

On resetting the eMIOS, the Unified Channels enter GPIO input mode.

37.7.1 Considerations

Before changing an operating mode, the UC must be programmed to GPIO mode and EMIOSA[n] and EMIOSB[n] registers must be updated with the correct values for the next operating mode. Then the EMIOSC[n] register can be written with the new operating mode. If a UC is changed from one mode to another without performing this procedure, the first operation cycle of the selected time base can be random, i.e., matches

can occur in random time if the contents of EMIOSA[n] or EMIOSB[n] were not updated with the correct value before the time base matches the previous contents of EMIOSA[n] or EMIOSB[n].

When interrupts are enabled, the software must clear the FLAG bits before exiting the interrupt service routine.

37.7.2 Application Information

Correlated output signals can be generated by all output operation modes. Bits OU[n] of eMIOS_OUDIS register can be used to control the update of these output signals.

In order to guarantee that the internal counters of correlated channels are incremented in the same clock cycle, the internal prescalers must be set up before enabling the global prescaler. If the internal prescalers are set after enabling the global prescaler, the internal counters may increment in the same ratio, but at a different clock cycle.

It is recommended to drive Output Disable Input signals with the emios_flag_out signals of some UCs running in SAIC mode. When an output disable condition happens, the software interrupt routine must service the output channels before servicing the channels running SAIC. This procedure avoid glitches in the output pins.

37.7.2.1 Time Base Generation

For MC and OPWFM with internal clock source operation modes, the internal counter rate can be modified by configuring the clock prescaler ratio. Figure 37-43 shows an example of a time base with prescaler ratio equal to one.

Note

MCB and OPWFMB modes have a different behavior.

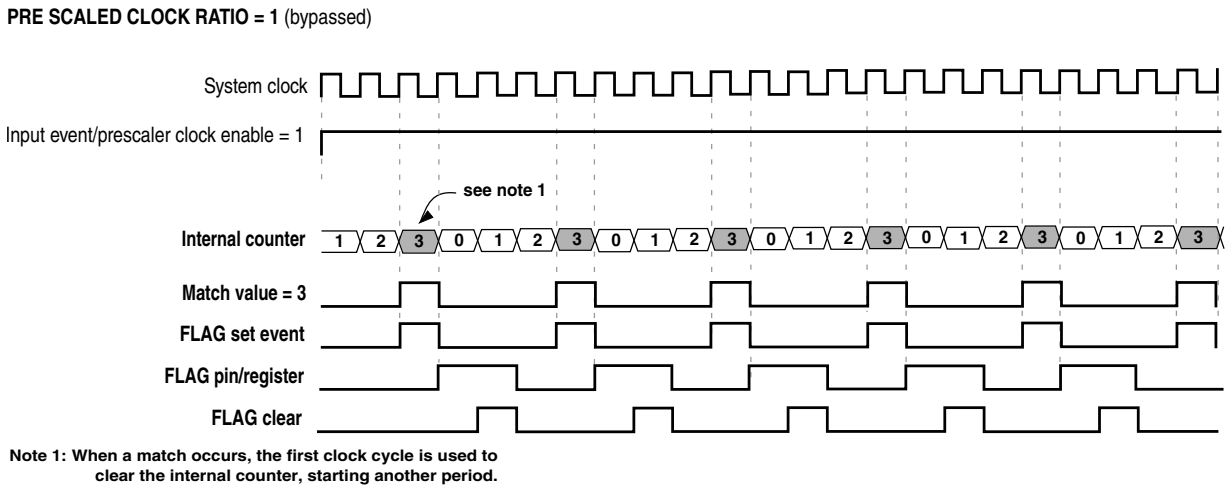


Figure 37-43. Time base period when running in the fastest prescaler ratio

If the prescaler ratio is greater than one or external clock is selected, the counter may behave in one of four different ways depending on the channel mode:

- If MC mode and Clear on Match Start and External Clock source are selected the internal counter behaves as described in Figure 37-44 .
- If MC mode and Clear on Match Start and Internal Clock source are selected the internal counter behaves as described in Figure 37-45.
- If MC mode and Clear on Match End are selected the internal counter behaves as described in Figure 37-46.
- If OPWFM mode is selected the internal counter behaves as described in Figure 37-45. The internal counter clears at the start of the match signal, skips the next prescaled clock edge and then increments in the subsequent prescaled clock edge.

Note

MCB and OPWFMB modes have a different behavior.

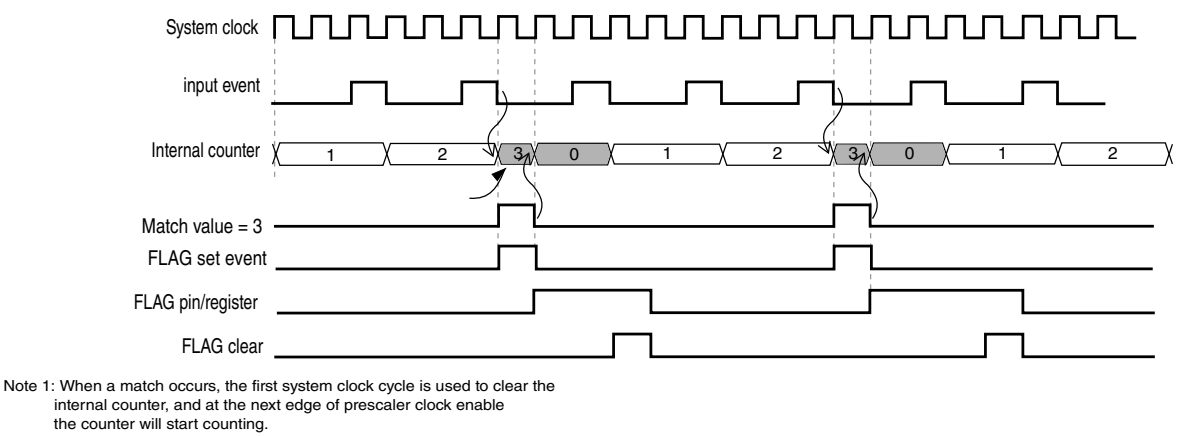
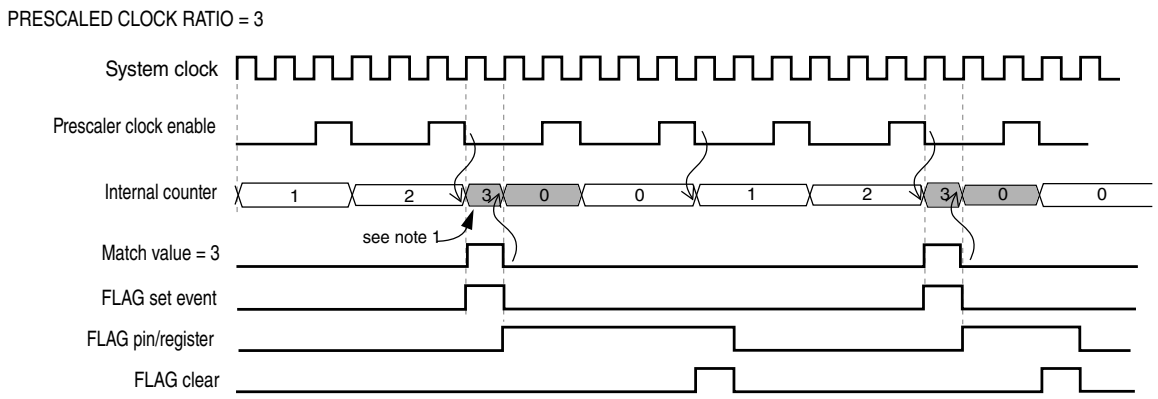
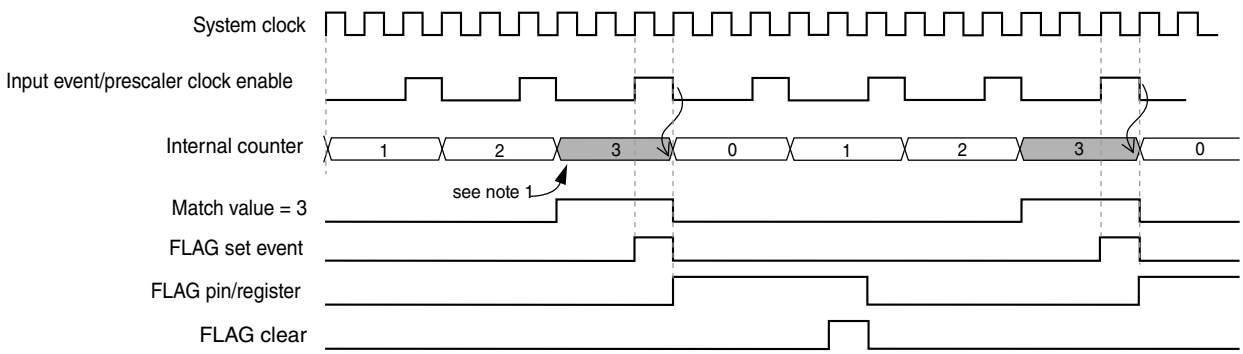


Figure 37-44. Time base generation with external clock and clear on match start



Note 1: When a match occurs, the first clock cycle is used to clear the internal counter, and only after a second edge of pre scaled clock the counter will start counting.

Figure 37-45. Time base generation with internal clock and clear on match start



Note 1: The match occurs only when the input event/prescaler clock enable is active. Then, the internal counter is immediately cleared.

Figure 37-46. Time base generation with clear on match end

37.7.2.2 Coherent Accesses

Reading the EMIOSA[n] register again in the same period of the last read of EMIOSB[n] register may lead to incoherent results. This will occur if the last read of EMIOSB[n] register occurred after a disabled B2 to B1 transfer.

37.7.2.3 Channel/Modes Initialization

The following basic steps summarize basic output mode startup, assuming the channels are initially in GPIO mode:

1. *[global]* Disable Global Prescaler.
2. *[timebase channel]* Disable Channel Prescaler.

3. *[timebase channel]* Write initial value at internal counter.
4. *[timebase channel]* Set A/B register.
5. *[timebase channel]* Set channel to MC(B) Up mode.
6. *[timebase channel]* Set prescaler ratio.
7. *[timebase channel]* Enable Channel Prescaler.
8. *[output channel]* Disable Channel Prescaler.
9. *[output channel]* Set A/B register.
10. *[output channel]* Select timebase input through BSL[1:0] bits.
11. *[output channel]* Enter output mode.
12. *[output channel]* Set prescaler ratio (same ratio as timebase channel).
13. *[output channel]* Enable Channel Prescaler.
14. *[global]* Enable Global Prescaler.
15. *[global]* Enable Global Time Base.

The timebase channel and the output channel may be the same for some applications such as in OPWFM(B) mode or whenever the output channel is intended to run the timebase itself.

At any time the flags can be configured.

Chapter 38
Power Control Unit (MC_PCU)

38.1 Introduction

38.1.1 Overview

The power control unit (MC_PCU) in conjunction with LPU_CTL and MC_ME, is used to reduce the overall chip power consumption. Power can be saved by disconnecting parts of the chip from the power supply via a power switching chip. The chip is grouped into multiple parts having this capability which are called “power domains”.

When a power domain is disconnected from the supply, the power consumption is reduced to zero in that domain. Any status information of such a power domain is lost. When re-connecting a power domain to the supply voltage, the domain draws an increased current until the power domain reaches its operational voltage.

Power domains are controlled on a chip mode basis. For each mode, software can configure whether a power domain is connected to the supply voltage (power-up state) or disconnected (power-down state). Maximum power saving is reached by entering the STANDBY0 mode.

The power-up/down sequences are handled by finite state machines to ensure a smooth and safe transition from one power state to the other.

Exiting the STANDBY0 mode can only be done via a system wakeup event as all power domains other than power domain #0 are in the power-down state.

In addition, the MC_PCU acts as a bridge for mapping the mapped peripheral peripheral to the MC_PCU address space.

The following figure depicts the MC_PCU block diagram.

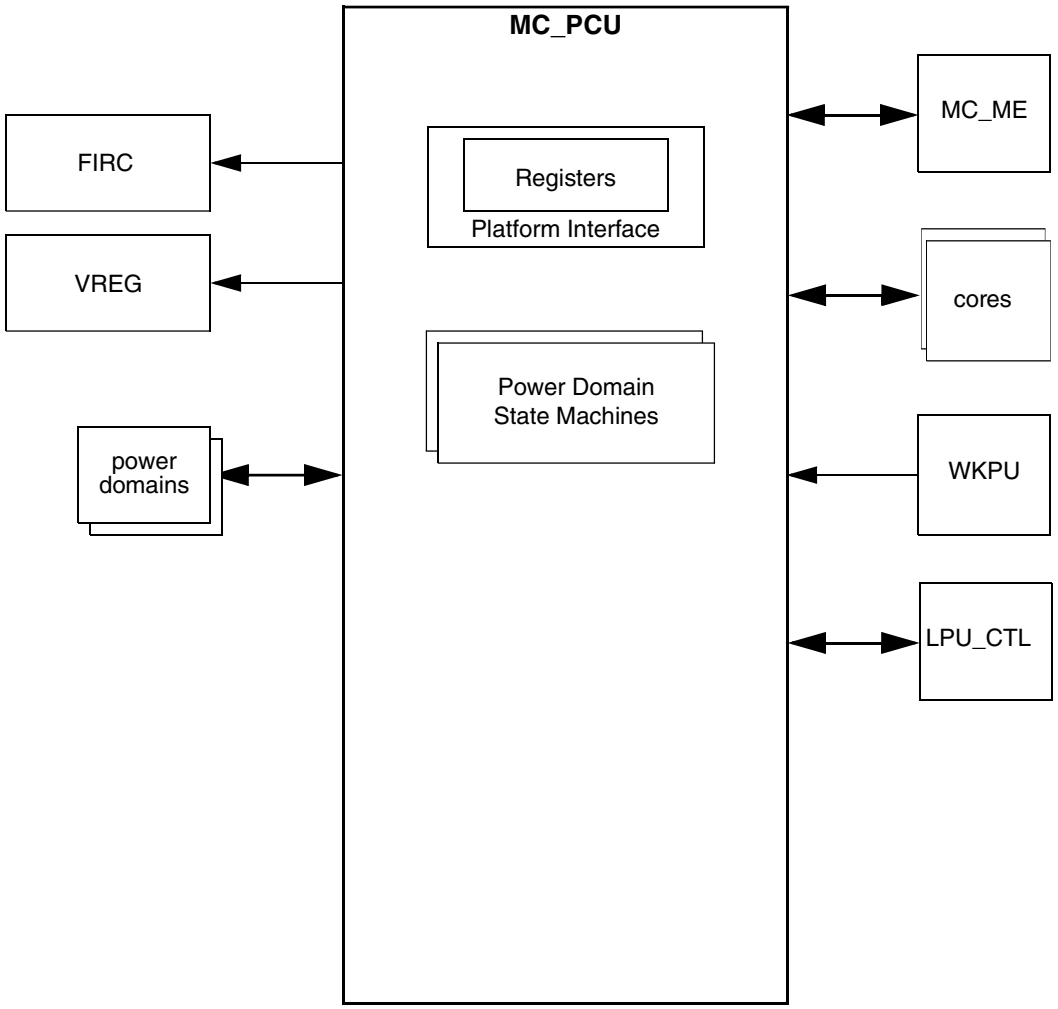


Figure 38-1. MC_PCU Block Diagram

38.1.2 Features

The MC_PCU includes the following features:

- PMCDIG registers are mapped to its address range.
- Bridge between MC_ME, LPU_CTL and PMCDIG to handle mode transitions from RUN to LPU mode, LPU_RUN to LPU_STANDBY and RUN to STANDBY0 mode.
- A handshake mechanism for power state changes thus guaranteeing operable voltage.

Chapter 39
Power Management Controller digital interface (PMCDIG)

39.1 Overview

The PMC has main linear voltage regulator that generates the internal 1.25 V digital supply from an external 3.3 V or 5 V supply. It also contains an option to bypass this main regulator and instead use an externally supplied 1.25 V inputIn addition, there are low power (LP) and ultra-low power (ULP) regulators that can be active, regardless of how the main 1.25 V is being supplied. Also, there is a linear regulator to power 3.3 V to the Flash HV domain (FLASHREG). Here, there is also an option to bypass and use an externally supplied 3.3 V input when device is in 3.3 V mode.

The PMC digital block generates a transfer error event to the system if the selected address is greater than the highest address in the PMC digital memory map. A transfer error is also generated if an unused address within the PMC digital memory space is accessed.

39.2 Features

The PMC provides the following main features:

- Compatible with 3.3 V and 5 V operation.
- No specific power sequencing required and all power domains guaranteed safe-state during power-up, power-down.
- POR circuitry
  - Guarantees safe initialization of MC.
- Precision voltage monitors
  - PMC voltage supply - POR (Power on Reset).
  - VDD core voltage supply - POR, HVD (high voltage detector), LVD (low voltage detector).

- Flash memory voltage supply - LVD.
- VDD_HV_A voltage supply - LVD.

- Bandgap voltage reference for
  - Voltage monitors and regulators.
  - ADC voltage reference channel.

39.3 Memory Map and Registers

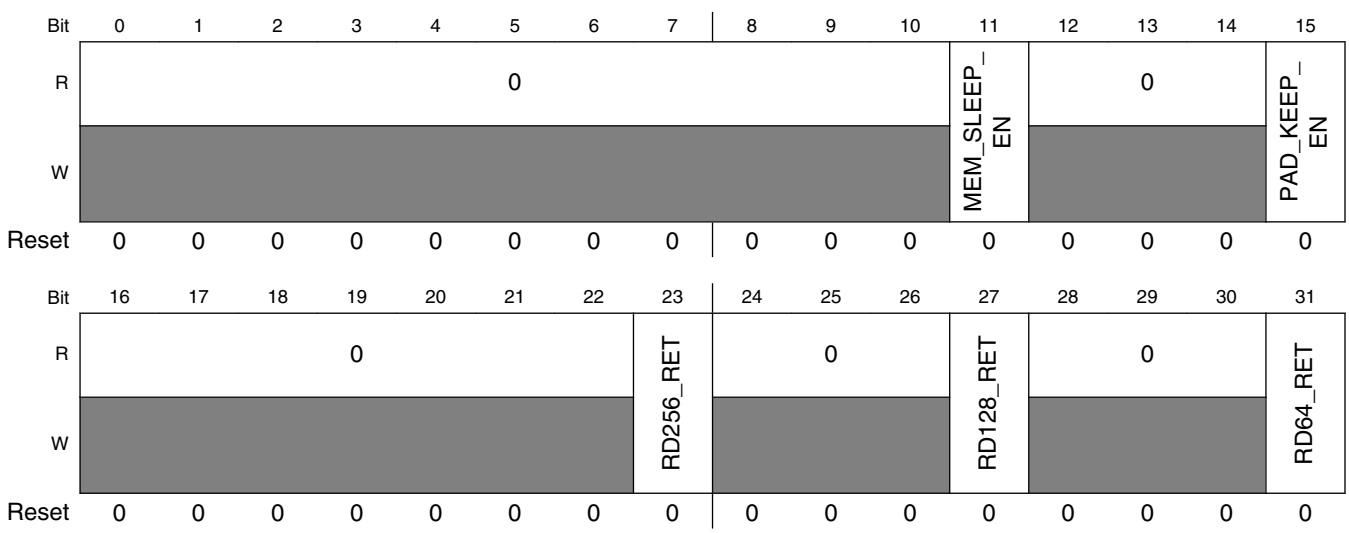
The following table shows the PMC memory map.

PMCDIG memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	RAM Domain Configuration Register (PMCDIG_RDCR)	32	R/W	0000_0000h	39.3.1/1042
4	Supply Gauge Status Register (PMCDIG_SGSR)	32	w1c	0000_0000h	39.3.2/1043
10	MISC Control Register (PMCDIG_MCR)	32	R/W	0000_0000h	39.3.3/1047

39.3.1 RAM Domain Configuration Register (PMCDIG_RDCR)

Address: 0h base + 0h offset = 0h



PMCDIG_RDCR field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PMCDIG_RDCR field descriptions (continued)

Field	Description
11 MEM_SLEEP_EN	Memory Sleep Enable When set, will enable the Sleep state for memories. Program this bit in STOP/LPU_STOP to take advantage of memory sleep state.
12–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 PAD_KEEP_EN	PAD Keeper Enable Enables the PAD keeping functionality where some Standby PADs can retain their configuration while in Power gated Modes (Standby). 0 Disable PAD Keeping functionality 1 Enabled Pad Keeping Functionality.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 RD256_RET	RAM Domain 256K Retention Retains the whole 256K RAM while in Standby Mode. 0 Switch to RAM supplies Opened (RAM cut is shut off)(default reset value). 1 Switch to RAM supplies Closed .
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 RD128_RET	RAM Domain 128K Retention Retains 128K RAM while in Standby Mode. 0 Switch to RAM supplies Opened (RAM cut is shut off) (default reset value). 1 Switch to RAM supplies Closed.
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RD64_RET	RAM Domain 64K Retention Retains 64K RAM while in Standby Mode. **NOTE:** If 8k of RAM is to be retained, it is implicitly achieved by disabling all bits here. 0 Switch to RAM supplies Opened (RAM cut is shut off) (default reset value). 1 Switch to RAM supplies Closed.

39.3.2 Supply Gauge Status Register (PMCDIG_SGSR)

This register is Reset only on a POR event.

On reading the flag indicating that a POR occured as 1 (after a POR event), the user should ideally ignore the rest of the bits in this register until a clear operation and read back is performed.

NOTE

On first Power up, bits from 19-31 may be set. It is suggested to read these bits and then write 1 to clear accordingly. In low power mode, if any of the LVD occurs then the behavior of this register will be similar to first power up.

NOTE

Bits 24-30 will remain set and will not get cleared if respective supply is below voltage monitor threshold. Please refer device datasheet for voltage monitor threshold levels.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			LP_STATUS		0	0	0	LVD_FLASH	LVD_IO_A_HI	LVD_IO_A_LO	LVD_LV_PD0_hot	LVD_LV_PD1_hot	LVD_LV_PD2_COLD	LVD_LV_PD2_HOT	POR_LV
W				w1c					w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMCDIG_SGSR field descriptions

Field	Description
0–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–20 LP_STATUS	Low Power Status Indicates the last Low Power Mode entered. Software needs to ensure that to effectively use these bits, the field should be cleared before making a low power transition and then read back once the read is possible again (RUN/LPU_RUN). 00 No LP Mode Transition Triggered OR cleared by user for new status to be captured. 01 Invalid state (will not occur and user should ensure that the registers are not cleared in a manner that results in the bits reflecting this value, USER should ALWAYS clear both bits together or just bit LP_STATUS[0]). 10 LPU mode 11 STANDBY mode
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 LVD_FLASH	LVD VDD_HV_ FLA Flag 0 Indicates that a Flash LVD event has not occured since last clear of the Flag. 1 Indicates that a Flash LVD event had occured in the past after the time instance when Flag was cleared.
25 LVD_IO_A_HI	LVD IO Segment A High Flag 0 Indicates that an IO_A_HI LVD event has not occured since last clear of the Flag. 1 Indicates that an IO_A_HI LVD event had occured in the past after the time instance when Flag was cleared.
26 LVD_IO_A_LO	LVD IO Segment A Low Flag 0 Indicates that an IO_A_LO LVD event has not occured since last clear of the Flag. 1 Indicates that an IO_A_LO LVD event had occured in the past after the time instance when Flag was cleared.
27 LVD_LV_PD0_ hot	LVD VDD_LV PD0 hot point Flag 0 Indicates that an LVD event on PD0 has not occured since last clear of the Flag. 1 Indicates that an LVD event on PD0 had occured in the past after the time instance when Flag was cleared.
28 LVD_LV_PD1_ hot	LVD VDD_LV PD1 hot point Flag 0 Indicates that an LVD event on PD1 has not occured since last clear of the Flag. 1 Indicates that an LVD event on PD1 had occured in the past after the time instance when Flag was cleared.
29 LVD_LV_PD2_ COLD	LVD_LV_PD2 Cold Point Flag 0 Indicates that an LVD event on PD2 COLD Point has not occured since last clear of the Flag. 1 Indicates that an LVD event on PD2 COLD Point had occured in the past after the time instance when Flag was cleared.

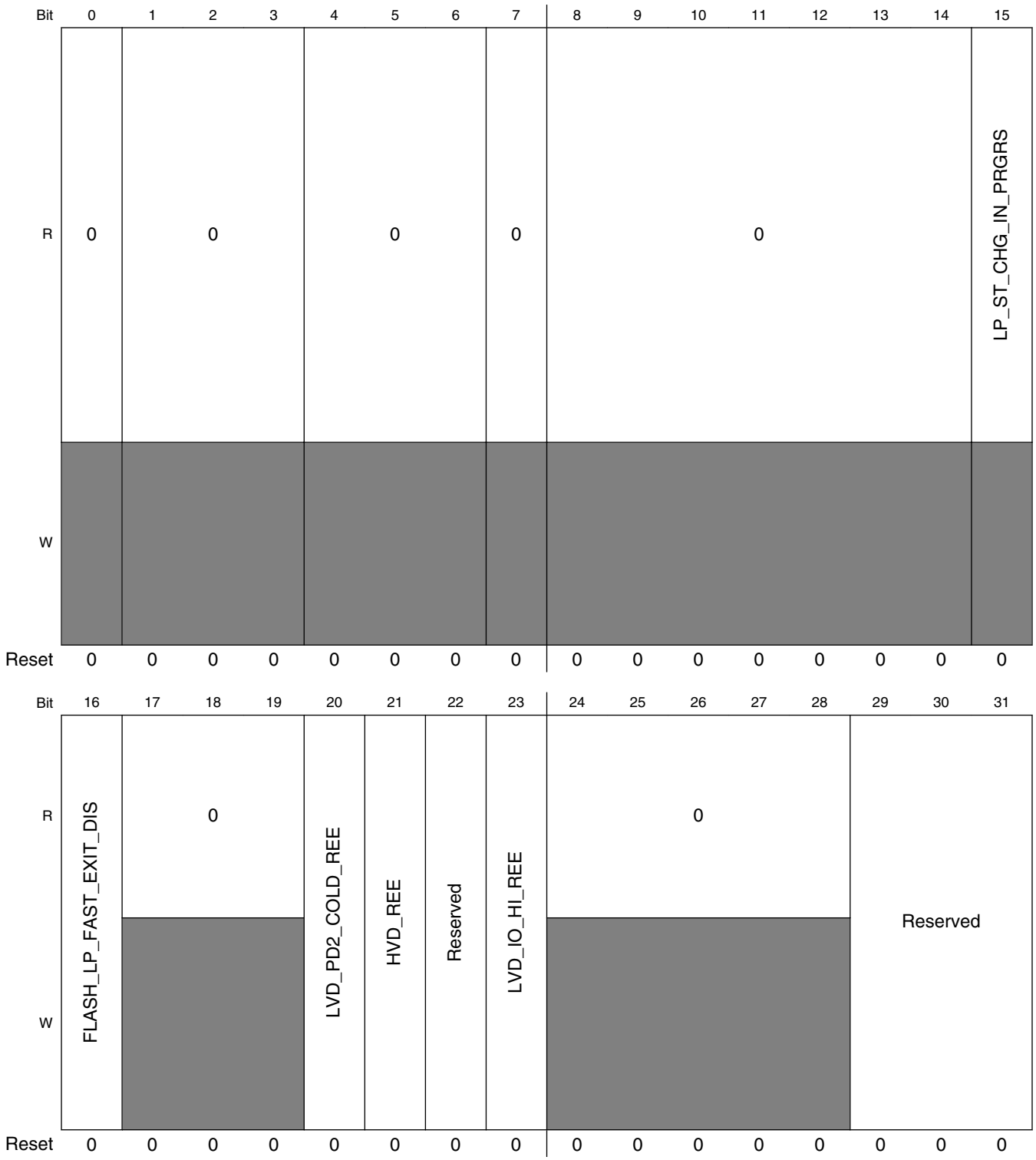
Table continues on the next page...

PMCDIG_SGSR field descriptions (continued)

Field	Description
30 LVD_LV_PD2_ HOT	LVD LV_PD2 hot Point Flag 0 Indicates that an LVD event on PD2 HOT Point has not occured since last clear of the Flag. 1 Indicates that an LVD event on PD2 HOT Point had occured in the past after the time instance when Flag was cleared.
31 POR_LV	Power On Reset Flag Flag Indicating that a Power On Reset occured. **NOTE:** To determine if the source of a power on destructive reset is por_lv, PORST or an essential LVD read PMC flags after reading the VOR bit. If none is set, it is PORST, else, the corresponding voltage reset. 0 Indicates that a POR event has not occured since last clear of the Flag. 1 Indicates that a POR event had occured in the past after the time instance when flag was cleared.

39.3.3 MISC Control Register (PMCDIG_MCR)

Address: 0h base + 10h offset = 10h



PMCDIG_MCR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 LP_ST_CHG_IN_PRGRS	Low Power State Change in Progress This flag indicates that the PMC is currently undergoing a low power mode state transition, if set. This bit should always be set to '0' before a low power mode transition request is initiated.
16 FLASH_LP_FAST_EXIT_DIS	Flash Low Power Fast Exit Disable When set, the Flash will not be enabled for fast wakeup sequences. While fast wakeup provides around 4% of wakeup time reduction from low power modes, this mode can generate Flash Low Voltage Detection (LVD) events due to Flash load transients and therefore it is recommended to disable it. **NOTE:** It is recommended to set this field to prevent false Flash LVD detections. 0 Flash Low Power Fast Exit Enabled 1 Flash Low Power Fast Exit Disabled
17–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 LVD_PD2_COLD_REE	LVD_PD2 Cold Point Flag Reset Event Enable Enables the LVD_PD2 Cold Point Flag functional reset source to RGM if set.
21 HVD_REE	HVD Reset Event Enable Enables the HVD functional reset source to RGM if set.
22 Reserved	This field is reserved. **NOTE:** These bits should not be written by the user. Writes to a reserved location can result in unpredictable functionality or behavior.
23 LVD_IO_HI_REE	LVD_IO_HI Reset Event Enable Enables the LVD_IO_HI functional reset source to RGM if set.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 Reserved	This field is reserved. **NOTE:** These bits should not be written by the user. Writes to a reserved location can result in unpredictable functionality or behavior.

39.4 Functional description

39.4.1 Analog PMC interface

The outputs from the PMC digital to the PMC analog circuit consists of enables for LVDs and the controls of the FPREG,FPREG_int LPREG, ULPREG , and FLASHREG, based on the power modes programmed in MC_ME and LPU_CTL.

The signals from the PMC analog to the PMC digital logic consists of simply POR and LVD event indication signals (both active high and active low, but only active low are used). These signals are used by the PMC digital logic to create the appropriate resets for the rest of the system.

39.4.2 Power-up Sequence

The following section describe the power sequence and the relation between the different supply during power-up and power-down.

NOTE

Refer to the device Datasheet for specific points on power sequence requirements relating to some of the supplies, if any.

Chapter 40
Mode Entry Module (MC_ME)

40.1 Introduction

40.1.1 Overview

The MC_ME controls the chip mode and mode transition sequences in all functional states. It also contains configuration, control and status registers accessible for the application.

The purpose of the MC_ME is to minimize the current consumption of device modules by:

- switching clocks on/off
- switching clock sources on/off
- switching power on/off to most of the device
- enabling system clock progressive switching when changing to a different power level configuration

The following figure shows the MC_ME block diagram.

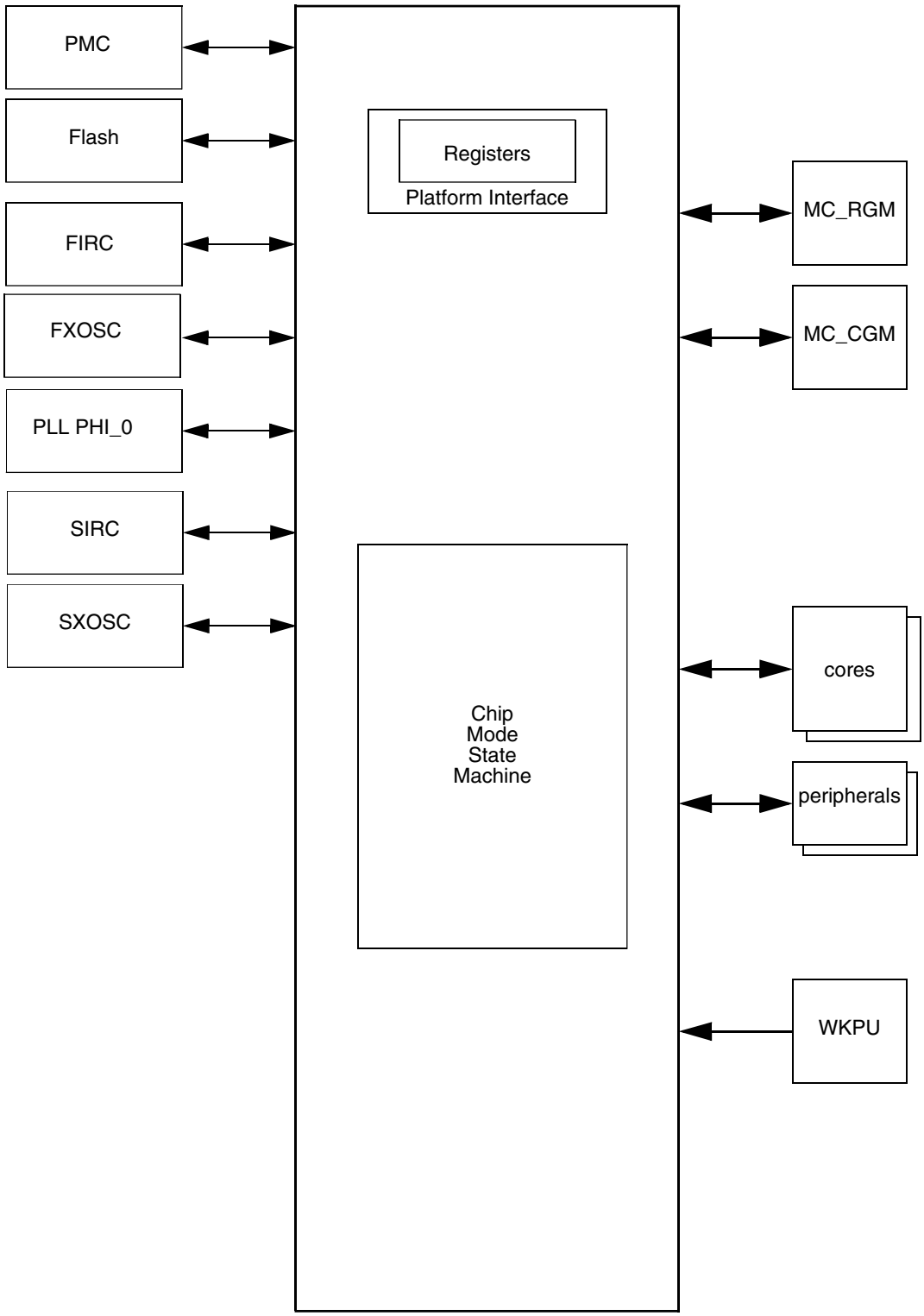


Figure 40-1. MC_ME Block Diagram

40.1.2 Features

The MC_ME includes the following features:

- control of the available modes by the ME_ME register
- definition of various chip mode configurations by the ME_<mode>_MC registers
- control of the actual chip mode by the ME_MCTL register for

  -switching between software-running modes

  -switching from software-running modes to low-power modes

  -switching to a mode to run diagnostics after a failure has been detected

  -triggering a device ‘functional’ reset which does not cause a start-up self test to run

  -triggering a device ‘destructive’ reset which may cause a start-up self test to run
- capture of the current mode and various resource status within the contents of the ME_GS register
- optional generation of various mode transition interrupts
- status bits for each cause of invalid mode transitions
- peripheral clock gating control based on the ME_RUN_PC0…7, ME_LP_PC0…7, and ME_PCTLn registers
- capture of current peripheral and additional core clock gated/enabled status
- core clock gating and boot control based on the ME_CCTLn and ME_CADDRn registers
- progressive system clock switching when transitioning from a lower power consumption mode to a higher power consumption mode, and vice versa
- capture of current core clock gated/enabled status

40.1.3 Use Cases

The following simplified use cases illustrate how the MC_ME module implements power saving mechanisms

40.1.3.1 Use Case 1: Software Tasks with Different Clock Requirements

Software runs one of the following three tasks at a time:

- Analog Monitor task

  — peripherals required: analog-to-digital converter

  — clock sources required: only 16 MHz internal RC oscillator, used for system clock and peripheral clocking

  — software: reads analog input; if the voltage meets a criteria, the Communications task is called

- Communication task

  — peripherals required: two CAN modules

  — clock sources required: only 8-40 MHz crystal oscillator, used for system clock and peripheral clocking

  — software: transmits analog data; if the response is positive, the Whole Chip task is activated

- Whole Chip task

  — peripherals required: all peripherals will be needed

  — clock sources required: PLL_PHI_0 will be used (requires 8-40 MHz crystal oscillator) for system clock and peripheral clocking

  — software: processes messages and information, then returns to Analog Monitor task

NOTE

All ME_PCTLx controlled peripherals should be accessed only after its clocks are enabled through ME_PCTLx and ME_RUN_PCx register configuration.

The three tasks are summarized in Table 40-1. Typically, many functions will execute (run) in a mode. The parameters for these modes are called Mode Configuration. The MC_ME implements a mode configuration register for each mode that specifies which clock sources are enabled, the system clock source, and other parameters.

After reset, code executes in a default run mode, called DRUN. Software can reconfigure DRUN and/or configure up to four RUNx modes, three of which are used here.

Each peripheral needs a clock for some modes, but not others. The combination of modes that have a clock enabled to a particular peripheral is called the Peripheral Configuration.

Table 40-1. MC_ME Use Case 1 Requirement Summary

Run Mode (tasks using that mode)	System Clock Selection	Clock Sources Required			Peripheral(s) Requiring Enabled Clock
		FIRC	FXOSC	PLL PHI_0	
	<------Mode Configuration---->				<--Peripheral Configuration--->
RUN0 (Analog Monitor)	FIRC	X			ADC_1
RUN1 (Communication)	FXOSC		X		FLEXCAN1, FLEXCAN2
RUN2 (Whole Chip)	PLL PHI_0		X	X	all peripherals

The table below reorganizes Table 40-1 for peripheral configuration to illustrate how the MC_ME implements controlling clocks to peripherals

Table 40-2. MC_ME Use Case 1 Peripheral Clocking Requirement Summary

Peripheral(s)	Modes with Enabled Peripheral Clock					Run Mode Peripheral Configuration	Peripheral's Control Register (specifies a run mode peripheral configuration)
	DRUN	RUN0	RUN1	RUN2	RUN3		
ADC_1		X		X		ME_RUN_PC1	ME_PCTL25[RUN_CFG] = 1
FLEXCAN 1, FLEXCAN 2			X	X		ME_RUN_PC2	ME_PCTL71[RUN_CFG] = 2, ME_PCTL72[RUN_CFG] = 2,
all other peripherals (with ME_PCTLx registers)				X		ME_RUN_PC3	all other ME_PCTLx[RUN_CFG] = 3

Each module that can be turned on or off by the MC_ME has its own perpheral control register (ME_PCTLx) which selects a run mode peripheral configuration register (ME_RUN_PCx). These peripheral configuration registers have bit fields for each mode to indicate whether a clock is enabled in each mode for peripherals selecting that peripheral configuration register. A complete list of ME_PCTLx registers is shown in Table "MC_ME Register Description".

NOTE

ME_RUN_PC0 is the default configuration for all peripherals during run modes. Therefore, changing its content may inadvertently turn on all peripherals whose ME_PCTLx content has not yet been changed to use another run mode peripheral configuration.

40.1.3.2 Use Case 2: Software Runs and Invokes Multiple Low-Power Modes

Software runs continuously while the machine (for example, automobile) has the ignition turned on. When the ignition is turned off, software stops executing code and enters the low-power STOP0 mode where all the I/O pads are kept in the same state, but PLL, FXOSC, the flash, and most of the peripheral clocks are configured to be turned off.

If the ignition needs to remain off for longer duration,then there is also a option to enter the low power STANDBY0 mode where most of the I/O pads,PLL,FXOSC,FIRC,the flash and most of the peripherals are configured to power-down.

The ignition state is an input to the chip. If the ignition is turned back on at any time, the input transition wakes up the device from either low-power mode. Since more of the chip is “off” in STANDBY0 mode than in STOP0 mode, STANDBY0 mode will take longer to recover.

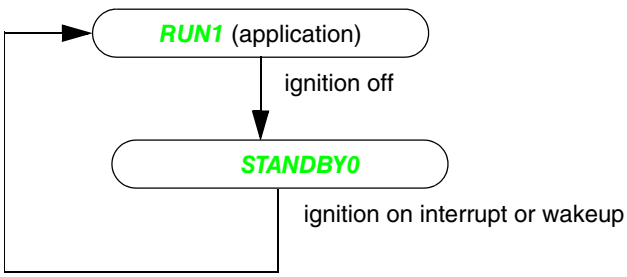


Figure 40-2. MC_ME Use Case 2: Low-Power Mode Flow Chart 1

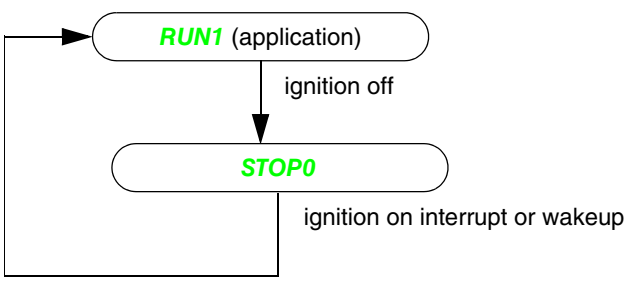


Figure 40-3. MC_ME Use Case 2: Low-Power Mode Flow Chart 2

After reset, software configures the chip to enter a run mode and two low-power modes:

- RUN1 mode
  - software runs continuously
  - if the ignition switch (a chip input) is switched off, software enters either STOP0 mode or STANDBY0 mode depending on Power Saving requirement.
- STOP0 mode
  - code does not run (no core clock)
  - all of the chip remains powered (no memory or register values lost)
  - STOP0 mode configuration:

        - pads are configured to maintain their state
        - PLL, FXOSC, the flash, and the main voltage regulator are configured to be off
    - STOP0 mode exits when
        - the ignition is detected via a wakeup input pin transition
- STANDBY0 mode
    - code does not run (core is power down)
    - Most of the chip remains power-down (memory or register values lost)
    - STANDBY0 mode configuration:
        - Selective pads are configured to maintain their state via padkeeping functionality.Most of the pads are powerdown.
        - PLL, FXOSC, the flash, and the main voltage regulator are configured to be powerdown.
    - STANDBY0 mode exits when
        - the ignition is detected via a wakeup input pin transition

The requirements for these modes are summarized in the following table:.

Table 40-3. MC_ME Use Case 2 Requirement Summary

Mode	System Clock Selection	Clock Sources			Memory Power Mode	Main Voltage Reg.	I/O Power Down Control	Peripherals Requiring Enabled Clock
		IRC	FXOSC	PLL	Flash			
Mode Configuration								
RUN1	PLL PHI_0	on	on	on	normal	on	off	all peripherals
STOP0	disabled	off	(off)	off	power-down	off	on	none
STANDBY0	disabled	off	(off)	disabled	power-down	off	off	none

The tables below summarize the modes where peripherals have a clock.

Table 40-4. MC_ME Use Case 2 Run Modes Peripheral Clocking Requirement

Peripheral	Run Modes with Enabled Peripheral Clock					Run Mode Peripheral Configuration	Peripheral's Control Register (specifies a run mode peripheral configuration)
	DRUN	RUN0	RUN1	RUN2	RUN3		
all peripherals (with ME_PCTLx registers)			X			ME_RUN_PC1	all ME_PCTLx[RUN_CFG] = 1

Table 40-5. MC_ME Use Case 2 Low-Power Modes Peripheral Clocking Requirement Summary

Peripheral	Low-Power Modes with Enabled Peripheral Clock		Low-Power Mode Peripheral Configuration	Peripheral's Control Register (specifies a run mode peripheral configuration)
	STOP0	STANDBY0		
all peripherals (with ME_PCTLx registers)			ME_LP_PC1	all ME_PCTLx[LP_CFG] = 1

40.1.3.3 Use Case 3: Mode Transitions Controlling Clocks to Multiple Cores

After initial boot, cores run or are disabled according to settings in their Core Control Register (ME_CCTLx).

Their boot address is contained in their Core Address Register (ME_CADDRx).The Core Address Register also has a configuration bit to reset the corresponding core on the next mode change where that core is configured to run (as defined in the Core Control Register), regardless of whether the core is currently running or is off.

NOTE

The fact that a core does not need to be on in the current mode for it to be reset using the ME_CADDRx[RMC] bit in the next mode transition allows the boot core, for example, to cause itself to re-boot from a new location when all other cores are off without needing to turn on one of the other cores to handle the mode transition.

These two registers are initialized by either

- the Boot Header (see the "BAF" chapter for details), or
- software with a mode transition to the configured mode

In the case of a transition between two RUNx modes where one or more cores will have their clocks gated off, software on a dedicated core whose clock will not be turned off must handle any necessary handshaking with the core(s) that will be losing their clock in the target mode.

An example of where multiple cores do not execute in all modes is shown in the following table. In this case, Z4 is the ‘master’ core that will handle the handshaking with the other cores.

Table 40-6. MC_ME Use Case 3 Run Modes Peripheral Clocking Requirement

Peripheral	Run Modes with Core Enabled Clock					Core's Control Register (specifies during which run mode the core run modes
	DRUN	RUN0	RUN1	RUN2	RUN3	
Z4 Core	X	X	X	X	X	ME_CCTL1
Z2 Core					X	ME_CCTL3

It is the responsibility of software to implement any needed handshaking when changing modes where a core loses a clock.

A high-level possible sequence is shown in the following figure for changing modes from RUN3 to RUN2. In this case, Z2 core, which is running in RUN3 mode, is turned off in RUN2 mode.

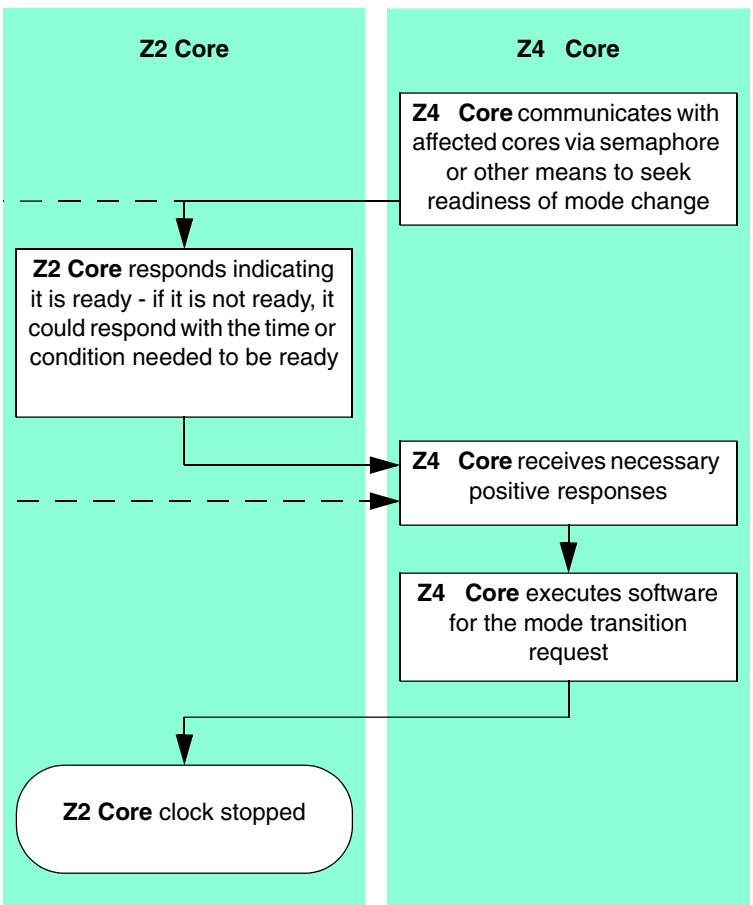


Figure 40-4. MC_ME Use Case 3: Possible mode Change sequence for turning off a core that is currently running (RUN3 to RUN2 in this case)

A high-level possible sequence is shown in the following figure for changing modes from DRUN to RUN3. In this case, Z4 Core, Z2 Core, all of which are off in DRUN mode, are turned on in RUN3 mode.

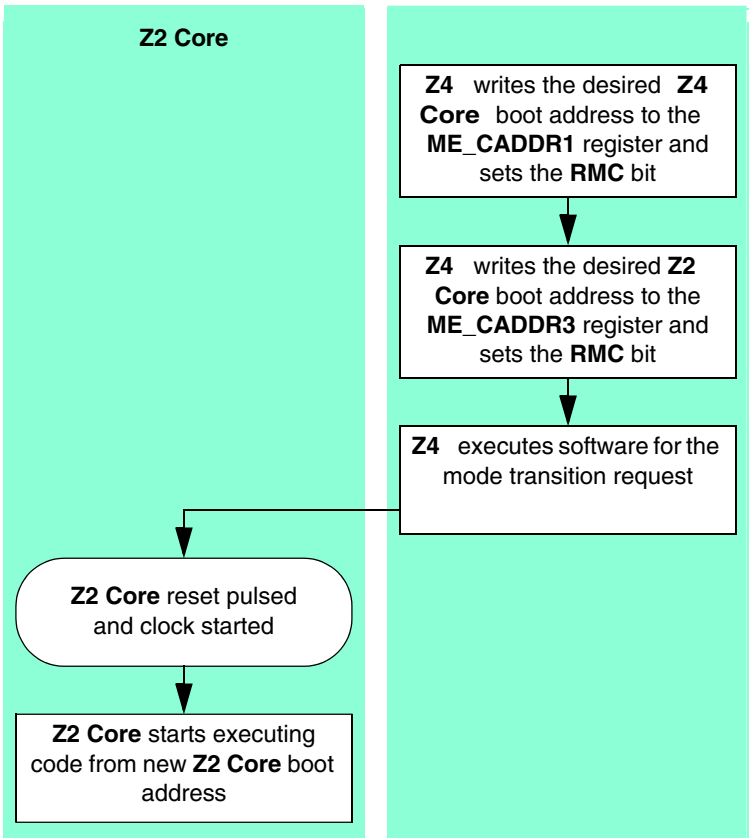


Figure 40-5. MC_ME Use Case 3: Possible mode change sequence for turning on one or more cores that are currently off (DRUN to RUN3 in this case)

40.1.4 Modes of Operation

The MC_ME is based on several chip modes corresponding to different usage models of the chip. Each mode is configurable and can define a policy for energy and processing power management to fit particular system requirements. An application can easily switch from one mode to another depending on the current needs of the system. The operating modes controlled by the MC_ME are divided into system and user modes. The system modes are modes such as RESET, DRUN, andSAFE. These modes aim to ease the configuration and monitoring of the system. The user modes are modes such as RUN0…3, and STOP0 which can be configured to meet the application requirements in terms of energy management and available processing power. The modes DRUN, SAFE, and RUN0…3 are the chip software running modes.

NOTE

AXBS masters cannot operate in SYSTEM STOP modes as system resources such as, DMA, SRAM, AXBS are clock gated. Also, NPC should be enabled if mode transition is initiated while debugger is connected.

The following table describes the MC_ME modes.

Table 40-7. MC_ME Mode Descriptions

Name	Description	Entry	Exit
RESET	This is a chip-wide virtual mode during which the application is not active. The system remains in this mode until all resources are available for the embedded software to take control of the chip. It manages hardware initialization of chip configuration, voltage regulators, clock sources, and flash modules.	system reset assertion from MC_RGM	system reset deassertion from MC_RGM
DRUN	This is the entry mode for the embedded software. It provides full accessibility to the system and enables the configuration of the system at startup. It provides the unique gate to enter user modes. BAM when present is executed in DRUN mode.	system reset deassertion from MC_RGM, software request from SAFE and RUN0…3	system reset assertion, RUN0…3, SAFE via software or hardware failure.
SAFE	This is a chip-wide service mode which may be entered on the detection of a recoverable error. It forces the system into a pre-defined safe configuration from which the system may try to recover.	hardware failure, software request from DRUN and RUN0…3	system reset assertion, DRUN via software
RUN0…3	These are software running modes where most processing activity is done. These various run modes allow to enable different clock & power configurations of the system with respect to each other.	software request from DRUN or other RUN0…3, interrupt event from interrupt or wakeup event from STOP0	system reset assertion, SAFE via software or hardware failure, DRUN and other RUN0…3 modes, STOP0 via software
STOP0	This is an advanced low-power mode during which the clock to the core is disabled. It may be configured to switch off most of the peripherals including clock sources for efficient power management at the cost of higher wakeup latency.	software request from RUN0…3	system reset assertion, SAFE on hardware failure, RUN0…3 on interrupt event or wakeup event
STANDBY0	This is a reduced-leakage low-power mode during which power supply is cut off from most of the chip. Wakeup from this mode takes a relatively long time, and content is lost or must be restored from backup	software request from RUN0…3, DRUN modes	system reset assertion, DRUN on wakeup event

40.1.4.1 Interrupts handling during mode transitions

All peripherals that will be disabled in the target mode must have their interrupt flags cleared prior to target mode entry. Before entering the target mode, software must ensure that all interrupt flags are cleared for those peripheral that are programmed to be disabled in the target mode. A pending interrupt from these peripherals at target mode entry will block the mode transition or possibly lead to unspecified behavior.

40.1.4.2 Advanced Low power user modes

The device adds two further advanced low power user modes, which are described in detail within LPU, Modes of operation : LPU_RUN and LPU_STOP modes.

For more information see table,Table 2-9

40.2 External Signal Description

The MC_ME has no connections to any external pins.

40.3 Memory Map and Register Definition

The MC_ME contains registers for:

- mode selection and status reporting
- mode configuration
- mode transition interrupts status and mask control
- scalable number of peripheral sub-mode selection and status reporting
- scalable number of additional core enabling and disabling control and status reporting

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the ME_RUN_PC0 register may be accessed as a word at address 0x080, as a half-word at address 0x082, or as a byte at address 0x083.

Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behavior cannot be changed.

NOTE

All registers are read only in user mode.

MC_ME memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Global Status Register (MC_ME_GS)	32	R	0C13_0010h	40.3.1/1067
4	Mode Control Register (MC_ME_MCTL)	32	R/W	3000_A50Fh	40.3.2/1070
8	Mode Enable Register (MC_ME_ME)	32	R/W	0000_A5FDh	40.3.3/1072
C	Interrupt Status Register (MC_ME_IS)	32	w1c	0000_0000h	40.3.4/1074
10	Interrupt Mask Register (MC_ME_IM)	32	R/W	0000_0000h	40.3.5/1075
14	Invalid Mode Transition Status Register (MC_ME_IMTS)	32	w1c	0000_0000h	40.3.6/1077
18	Debug Mode Transition Status Register (MC_ME_DMTS)	32	R	0000_0000h	40.3.7/1079
20	RESET Mode Configuration Register (MC_ME_RESET_MC)	32	R	0013_0090h	40.3.8/1084

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
28	SAFE Mode Configuration Register (MC_ME_SAFE_MC)	32	R/W	0093_0010h	40.3.9/1086
2C	DRUN Mode Configuration Register (MC_ME_DRUN_MC)	32	R/W	See section	40.3.10/ 1088
30	RUN0 Mode Configuration Register (MC_ME_RUN0_MC)	32	R/W	0013_0010h	40.3.11/ 1091
34	RUN1 Mode Configuration Register (MC_ME_RUN1_MC)	32	R/W	0013_0010h	40.3.12/ 1093
38	RUN2 Mode Configuration Register (MC_ME_RUN2_MC)	32	R/W	0013_0010h	40.3.13/ 1096
3C	RUN3 Mode Configuration Register (MC_ME_RUN3_MC)	32	R/W	0013_0010h	40.3.14/ 1098
48	STOP0 Mode Configuration Register (MC_ME_STOP_MC)	32	R/W	0011_0010h	40.3.15/ 1100
54	STANDBY0 Mode Configuration Register (MC_ME_STANDBY_MC)	32	R/W	0081_001Fh	40.3.16/ 1103
60	Peripheral Status Register 0 (MC_ME_PS0)	32	R	0000_0000h	40.3.17/ 1105
64	Peripheral Status Register 1 (MC_ME_PS1)	32	R	0000_0000h	40.3.18/ 1108
68	Peripheral Status Register 2 (MC_ME_PS2)	32	R	See section	40.3.19/ 1110
6C	Peripheral Status Register 3 (MC_ME_PS3)	32	R	0000_0000h	40.3.20/ 1112
80	Run Peripheral Configuration Register (MC_ME_RUN_PC0)	32	R/W	0000_0000h	40.3.21/ 1114
84	Run Peripheral Configuration Register (MC_ME_RUN_PC1)	32	R/W	0000_0000h	40.3.21/ 1114
88	Run Peripheral Configuration Register (MC_ME_RUN_PC2)	32	R/W	0000_0000h	40.3.21/ 1114
8C	Run Peripheral Configuration Register (MC_ME_RUN_PC3)	32	R/W	0000_0000h	40.3.21/ 1114
90	Run Peripheral Configuration Register (MC_ME_RUN_PC4)	32	R/W	0000_0000h	40.3.21/ 1114
94	Run Peripheral Configuration Register (MC_ME_RUN_PC5)	32	R/W	0000_0000h	40.3.21/ 1114
98	Run Peripheral Configuration Register (MC_ME_RUN_PC6)	32	R/W	0000_0000h	40.3.21/ 1114
9C	Run Peripheral Configuration Register (MC_ME_RUN_PC7)	32	R/W	0000_0000h	40.3.21/ 1114
A0	Low-Power Peripheral Configuration Register (MC_ME_LP_PC0)	32	R/W	0000_0000h	40.3.22/ 1115
A4	Low-Power Peripheral Configuration Register (MC_ME_LP_PC1)	32	R/W	0000_0000h	40.3.22/ 1115
A8	Low-Power Peripheral Configuration Register (MC_ME_LP_PC2)	32	R/W	0000_0000h	40.3.22/ 1115

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AC	Low-Power Peripheral Configuration Register (MC_ME_LP_PC3)	32	R/W	0000_0000h	40.3.22/ 1115
B0	Low-Power Peripheral Configuration Register (MC_ME_LP_PC4)	32	R/W	0000_0000h	40.3.22/ 1115
B4	Low-Power Peripheral Configuration Register (MC_ME_LP_PC5)	32	R/W	0000_0000h	40.3.22/ 1115
B8	Low-Power Peripheral Configuration Register (MC_ME_LP_PC6)	32	R/W	0000_0000h	40.3.22/ 1115
BC	Low-Power Peripheral Configuration Register (MC_ME_LP_PC7)	32	R/W	0000_0000h	40.3.22/ 1115
C0	BCTU Peripheral Control Register (MC_ME_PCTL0)	8	R/W	00h	40.3.23/ 1116
C1	eMIOS_0 Peripheral Control Register (MC_ME_PCTL1)	8	R/W	00h	40.3.24/ 1117
C2	eMIOS_1 Peripheral Control Register (MC_ME_PCTL2)	8	R/W	00h	40.3.25/ 1118
CA	SAI0 Peripheral Control Register (MC_ME_PCTL10)	8	R/W	00h	40.3.26/ 1119
CB	SAI1 Peripheral Control Register (MC_ME_PCTL11)	8	R/W	00h	40.3.27/ 1120
CC	SAI2 Peripheral Control Register (MC_ME_PCTL12)	8	R/W	00h	40.3.28/ 1121
CF	ENET Peripheral Control Register (MC_ME_PCTL15)	8	R/W	00h	40.3.29/ 1122
D4	CMP_0 Peripheral Control Register (MC_ME_PCTL20)	8	R/W	00h	40.3.30/ 1123
D5	CMP_1 Peripheral Control Register (MC_ME_PCTL21)	8	R/W	00h	40.3.31/ 1124
D6	CMP_2 Peripheral Control Register (MC_ME_PCTL22)	8	R/W	00h	40.3.32/ 1125
D8	ADC_0 Peripheral Control Register (MC_ME_PCTL24)	8	R/W	00h	40.3.33/ 1126
D9	ADC_1 Peripheral Control Register (MC_ME_PCTL25)	8	R/W	00h	40.3.34/ 1127
DC	FlexRay Peripheral Control Register (MC_ME_PCTL28)	8	R/W	00h	40.3.35/ 1128
DE	IIC_0 Peripheral Control Register (MC_ME_PCTL30)	8	R/W	00h	40.3.36/ 1129
DF	IIC_1 Peripheral Control Register (MC_ME_PCTL31)	8	R/W	00h	40.3.37/ 1130
E0	IIC_2 Peripheral Control Register (MC_ME_PCTL32)	8	R/W	00h	40.3.38/ 1131
E1	IIC_3 Peripheral Control Register (MC_ME_PCTL33)	8	R/W	00h	40.3.39/ 1132

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E8	DSPI_0 Peripheral Control Register (MC_ME_PCTL40)	8	R/W	00h	40.3.40/ 1133
E9	DSPI_1 Peripheral Control Register (MC_ME_PCTL41)	8	R/W	00h	40.3.41/ 1134
EA	DSPI_2 Peripheral Control Register (MC_ME_PCTL42)	8	R/W	00h	40.3.42/ 1135
EB	DSPI_3 Peripheral Control Register (MC_ME_PCTL43)	8	R/W	00h	40.3.43/ 1136
F2	LIN_0 Peripheral Control Register (MC_ME_PCTL50)	8	R/W	00h	40.3.44/ 1137
F3	LIN_1 Peripheral Control Register (MC_ME_PCTL51)	8	R/W	00h	40.3.45/ 1138
F4	LIN_2 Peripheral Control Register (MC_ME_PCTL52)	8	R/W	00h	40.3.46/ 1139
F5	LIN_3 Peripheral Control Register (MC_ME_PCTL53)	8	R/W	00h	40.3.47/ 1140
F6	LIN_4 Peripheral Control Register (MC_ME_PCTL54)	8	R/W	00h	40.3.48/ 1141
F7	LIN_5 Peripheral Control Register (MC_ME_PCTL55)	8	R/W	00h	40.3.49/ 1142
F8	LIN_6 Peripheral Control Register (MC_ME_PCTL56)	8	R/W	00h	40.3.50/ 1143
F9	LIN_7 Peripheral Control Register (MC_ME_PCTL57)	8	R/W	00h	40.3.51/ 1144
FA	LIN 8 Peripheral Control Register (MC_ME_PCTL58)	8	R/W	00h	40.3.52/ 1145
FB	LIN_9 Peripheral Control Register (MC_ME_PCTL59)	8	R/W	00h	40.3.53/ 1146
FC	LIN_10 Peripheral Control Register (MC_ME_PCTL60)	8	R/W	00h	40.3.54/ 1147
FD	LIN_11 Peripheral Control Register (MC_ME_PCTL61)	8	R/W	00h	40.3.55/ 1148
FE	LIN 12 Peripheral Control Register (MC_ME_PCTL62)	8	R/W	00h	40.3.56/ 1149
FF	LIN_13 Peripheral Control Register (MC_ME_PCTL63)	8	R/W	00h	40.3.57/ 1150
100	LIN 14 Peripheral Control Register (MC_ME_PCTL64)	8	R/W	00h	40.3.58/ 1151
101	LIN_15 Peripheral Control Register (MC_ME_PCTL65)	8	R/W	00h	40.3.59/ 1152
106	FLEXCAN_0 Peripheral Control Register (MC_ME_PCTL70)	8	R/W	00h	40.3.60/ 1153
107	FLEXCAN_1 Peripheral Control Register (MC_ME_PCTL71)	8	R/W	00h	40.3.61/ 1154

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
108	FLEXCAN_2 Peripheral Control Register (MC_ME_PCTL72)	8	R/W	00h	40.3.62/ 1155
109	FLEXCAN_3 Peripheral Control Register (MC_ME_PCTL73)	8	R/W	00h	40.3.63/ 1156
10A	FLEXCAN_4 Peripheral Control Register (MC_ME_PCTL74)	8	R/W	00h	40.3.64/ 1157
10B	FLEXCAN_5 Peripheral Control Register (MC_ME_PCTL75)	8	R/W	00h	40.3.65/ 1158
10C	FLEXCAN_6 Peripheral Control Register (MC_ME_PCTL76)	8	R/W	00h	40.3.66/ 1159
10D	FLEXCAN_7 Peripheral Control Register (MC_ME_PCTL77)	8	R/W	00h	40.3.67/ 1160
112	JDC Control Register (MC_ME_PCTL82)	8	R/W	00h	40.3.68/ 1161
113	MEMU_0 Control Register (MC_ME_PCTL83)	8	R/W	00h	40.3.69/ 1162
114	CRC Control Register (MC_ME_PCTL84)	8	R/W	00h	40.3.70/ 1163
11A	DMAMUX Control Register (MC_ME_PCTL90)	8	R/W	00h	40.3.71/ 1164
11B	PIT_RTI_0 Control Register (MC_ME_PCTL91)	8	R/W	00h	40.3.72/ 1165
11D	WKPU Control Register (MC_ME_PCTL93)	8	R/W	00h	40.3.73/ 1166
120	SPI0 Control Register (MC_ME_PCTL96)	8	R/W	00h	40.3.74/ 1167
121	SPI1 Control Register (MC_ME_PCTL97)	8	R/W	00h	40.3.75/ 1168
122	SPI2 Control Register (MC_ME_PCTL98)	8	R/W	00h	40.3.76/ 1169
123	SPI3 Control Register (MC_ME_PCTL99)	8	R/W	00h	40.3.77/ 1170
126	RTC-API Control Register (MC_ME_PCTL102)	8	R/W	00h	40.3.78/ 1171
129	MEMU_1 Control Register (MC_ME_PCTL105)	8	R/W	00h	40.3.79/ 1172
1C0	Core Status Register (MC_ME_CS)	32	R	0000_0002h	40.3.80/ 1174
1C6	Core1 Control Register (MC_ME_CCTL1)	16	R	0018h	40.3.81/ 1175
1CA	Core3 Control Register (MC_ME_CCTL3)	16	R	See section	40.3.82/ 1176
1E4	CORE1 Address Register (MC_ME_CADDR1)	32	R/W	See section	40.3.83/ 1178

Table continues on the next page...

MC_ME memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1EC	CORE3 Address Register (MC_ME_CADDR3)	32	R/W	See section	40.3.84/ 1179

40.3.1 Global Status Register (MC_ME_GS)

This register contains global mode status.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_CURRENT_MODE				S_MTRANS	1	0		S_PDO	0		S_MVR	0		S_FLA	
W																
Reset	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							S_SXOSC	S_SIRCON	S_PLLON	S_FXOSC	S_FIRCON	S_SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

MC_ME_GS field descriptions

Field	Description
0–3 S_CURRENT_MODE	Current chip mode status 0000 RESET 0001 Reserved 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 Reserved 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 STANDBY0 1110 reserved 1111 reserved
4 S_MTRANS	Mode transition status 0 Mode transition process is not active 1 Mode transition is ongoing
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
6–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 S_PDO	Output power-down status — This bit specifies output power-down status of I/Os. This bit is asserted whenever outputs of pads are forced to high impedance state or the pads power sequence driver is switched off. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE mode, outputs of pads are forced to high impedance state and the pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional. In STANDBY0 mode, the power sequence driver and all pads except those mapped on wakeup lines are not powered and therefore high impedance. Wakeup lines configuration remains unchanged
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 S_MVR	Main voltage regulator status 0 Main voltage regulator is not ready 1 Main voltage regulator is ready for use
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 S_FLA	Flash availability status 00 Flash is not available 01 Flash is in power-down mode

Table continues on the next page...

MC_ME_GS field descriptions (continued)

Field	Description
	10 Flash is in power-down mode 11 Flash is in normal mode and available for use
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 S_SXOSC	32 KHz external oscillator status 0 32 KHz external oscillator is not stable 1 32 KHz external oscillator is providing a stable clock
24 S_SIRCON	128 KHz internal RC oscillator status 0 128 KHz internal RC oscillator is not stable 1 128 KHz internal RC oscillator is providing a stable clock
25 S_PLLON	PLL status 0 PLL is not stable 1 PLL is providing a stable clock
26 S_FXOSC	8-40 MHz crystal oscillator status **NOTE** S_XOSC alone should not be relied on. To get the current status of XOSC also check CMU0_ISR[OLRI] flag. XOSC is stable when MC_ME_GS[S_XOSC] is 1 and OLRI flag = 0 0 8-40 MHz crystal oscillator is not stable 1 8-40 MHz crystal oscillator is providing a stable clock
27 S_FIRCON	16 MHz internal RC oscillator status 0 16 MHz internal RC oscillator is not stable 1 16 MHz internal RC oscillator is providing a stable clock
28–31 S_SYSCLK	System clock switch status — These bits specify the system clock currently used by the system. 0000 16 MHz int. RC osc. 0001 FXOSC 0010 PLL PHI_0 0011 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled

40.3.2 Mode Control Register (MC_ME_MCTL)

This register is used to trigger software-controlled mode changes. Depending on the modes as enabled by ME_ME register bits, configurations corresponding to unavailable modes are reserved and access to ME_<mode>_MC registers must respect this for successful mode requests.

NOTE

Byte and half-word write accesses are not allowed for this register as a predefined key is required to change its value.

NOTE

Software should always check PMCDIG_MCR[LP_ST_CHG_IN_PRGRS] is 0 before a new mode transition request is initiated.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TARGET_MODE				0																											
W																	KEY															
Reset	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

MC_ME_MCTL field descriptions

Field	Description
0–3 TARGET_MODE	Target chip mode These bits provide the target chip mode to be entered by software programming. The mechanism to enter into any mode by software requires the write operation twice: first time with key, and second time with inverted key. These bits are automatically updated by hardware while entering SAFE on hardware request. Also, while exiting from the STOP0 modes on hardware exit events, these are updated with the appropriate RUN0…3 mode value. 0000 RESET (triggers a 'functional' reset event) 0001 Reserved 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 Reserved 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 STANDBY0

Table continues on the next page...

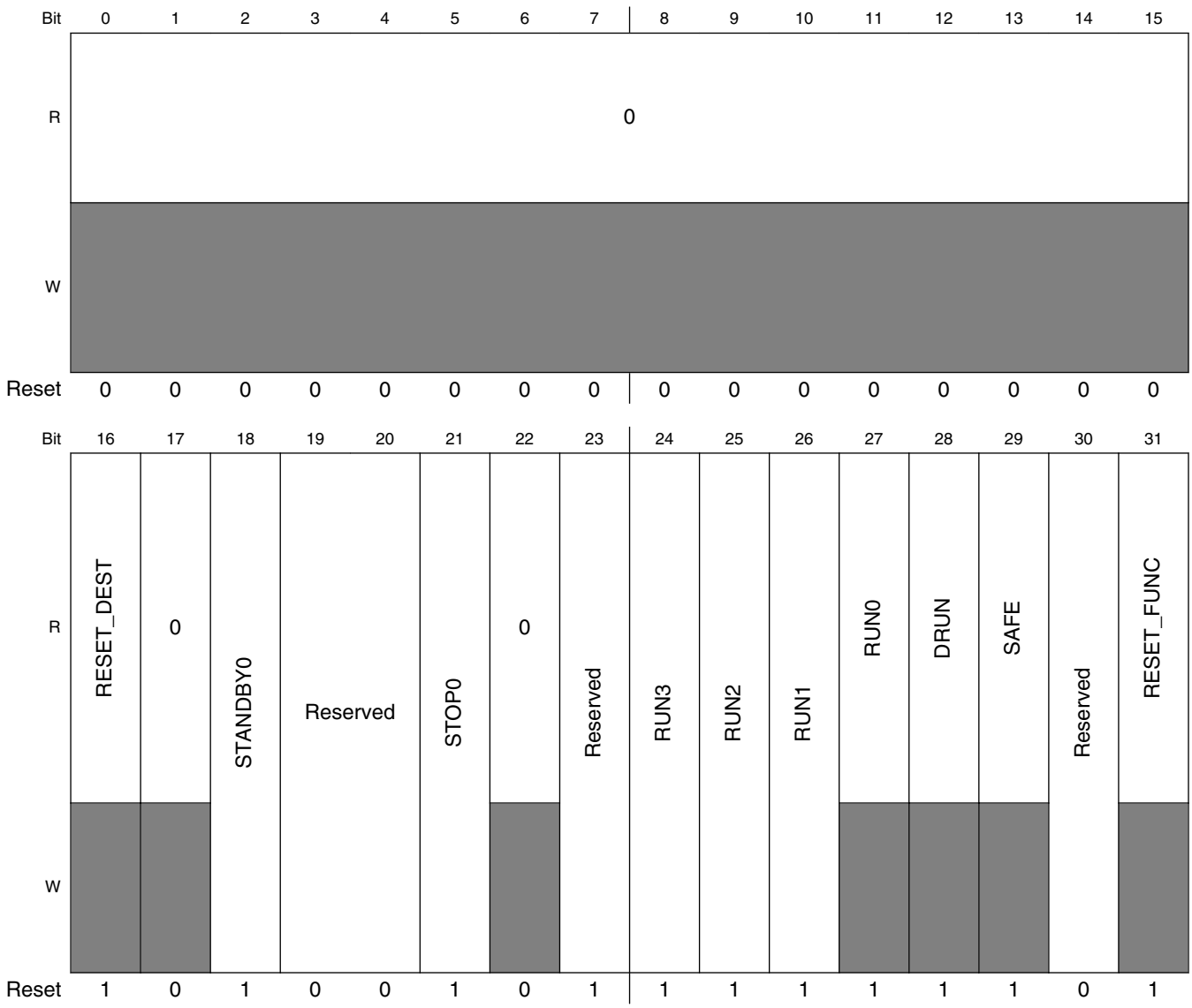
MC_ME_MCTL field descriptions (continued)

Field	Description
	1110 reserved 1111 RESET (triggers a ‘destructive’ reset event)
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 KEY	Control key — These bits enable write access to this register. Any write access to the register with a value different from the keys is ignored. Read access will always return inverted key. KEY: 0101101011110000 (0x5AF0) INVERTED KEY: 1010010100001111 (0xA50F)

40.3.3 Mode Enable Register (MC_ME_ME)

This register allows a way to disable the chip modes which are not required for a given chip. RESET_FUNC, SAFE, DRUN, RESET_DEST, and RUN0 modes are always enabled.

Address: 0h base + 8h offset = 8h



MC_ME_ME field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_ME field descriptions (continued)

Field	Description
16 RESET_DEST	‘destructive’ RESET mode enable 1 ‘destructive’ RESET mode is enabled
17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 STANDBY0	STANDBY0 mode enable 0 STANDBY0 mode is disabled 1 STANDBY0 mode is enabled
19–20 Reserved	This field is reserved.
21 STOP0	STOP0 mode enable 0 STOP0 mode is disabled 1 STOP0 mode is enabled
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved.
24 RUN3	RUN3 mode enable 0 RUN3 mode is disabled 1 RUN3 mode is enabled
25 RUN2	RUN2 mode enable 0 RUN2 mode is disabled 1 RUN2 mode is enabled
26 RUN1	RUN1 mode enable 0 RUN1 mode is disabled 1 RUN1 mode is enabled
27 RUN0	RUN0 mode enable 1 RUN0 mode is enabled
28 DRUN	DRUN mode enable 1 DRUN mode is enabled
29 SAFE	SAFE mode enable 1 SAFE mode is enabled
30 Reserved	This field is reserved.
31 RESET_FUNC	‘functional’ RESET mode enable 1 ‘functional’ RESET mode is enabled 0 ‘functional’ RESET mode is disabled

40.3.4 Interrupt Status Register (MC_ME_IS)

This register provides the current interrupt status.

NOTE

After BAF exits, and the control moves to application code, IS register is not in Default Reset State. It shows 0x1 which implies 'Mode transition complete interrupt is pending'.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										I_ICONF_CC	Reserved	I_ICONF	I_IMODE	I_SAFE	I_MTC
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_IS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 I_ICONF_CC	Invalid mode configuration interrupt (core configuration) — This bit is set if a write access to one of the ME_CADDRn registers is attempted while a mode transition is in progress. 0 No write to an ME_CADDRn register was attempted during an ongoing mode transition 1 A write to an ME_CADDRn register was attempted during an ongoing mode transition

Table continues on the next page...

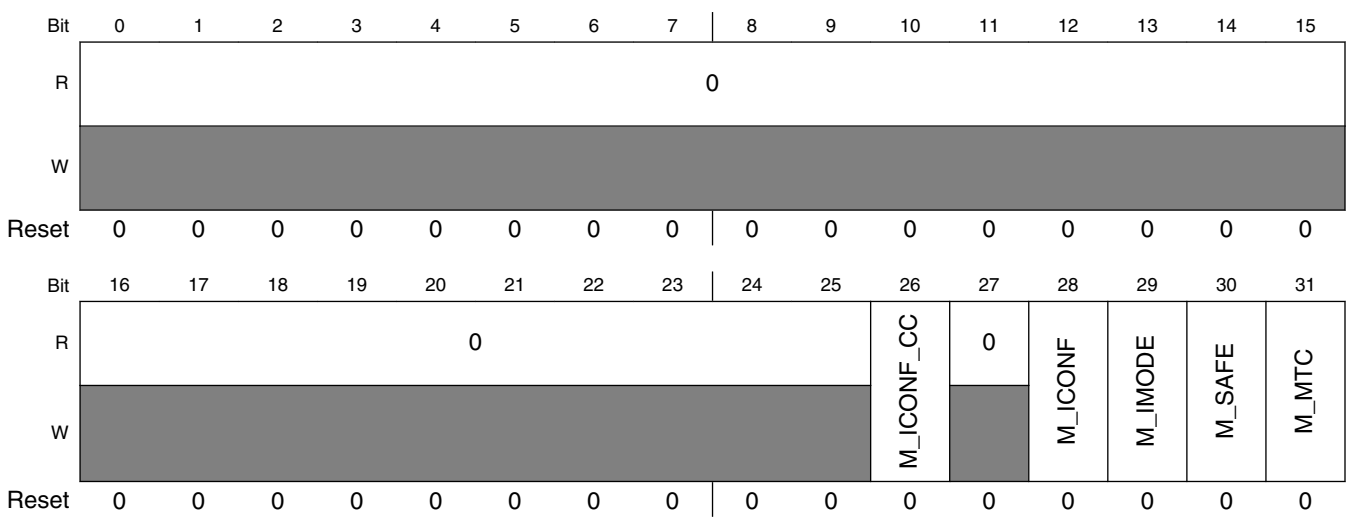
MC_ME_IS field descriptions (continued)

Field	Description
27 Reserved	This field is reserved and can be set to 1 by hardware during some mode transitions. The setting of this bit has no impact on the device. This field is reserved.
28 I_ICONF	Invalid mode configuration interrupt — This bit is set whenever a write operation to ME_<mode>_MC registers with invalid mode configuration is attempted. It is cleared by writing a '1' to this bit. 0 No invalid mode configuration interrupt occurred 1 Invalid mode configuration interrupt is pending
29 I_IMODE	Invalid mode interrupt — This bit is set whenever an invalid mode transition is requested. It is cleared by writing a '1' to this bit. 0 No invalid mode interrupt occurred 1 Invalid mode interrupt is pending
30 I_SAFE	SAFE mode interrupt — This bit is set whenever the chip enters SAFE mode on hardware requests generated in the system. It is cleared by writing a '1' to this bit. 0 No SAFE mode interrupt occurred 1 SAFE mode interrupt is pending
31 I_MTC	Mode transition complete interrupt — This bit is set whenever the mode transition process completes (S_MTRANS transits from 1 to 0). It is cleared by writing a '1' to this bit. This mode transition interrupt bit will not be set while entering low-power modes STOP0, or STANDBY0. 0 No mode transition complete interrupt occurred 1 Mode transition complete interrupt is pending

40.3.5 Interrupt Mask Register (MC_ME_IM)

This register controls whether an event generates an interrupt or not.

Address: 0h base + 10h offset = 10h



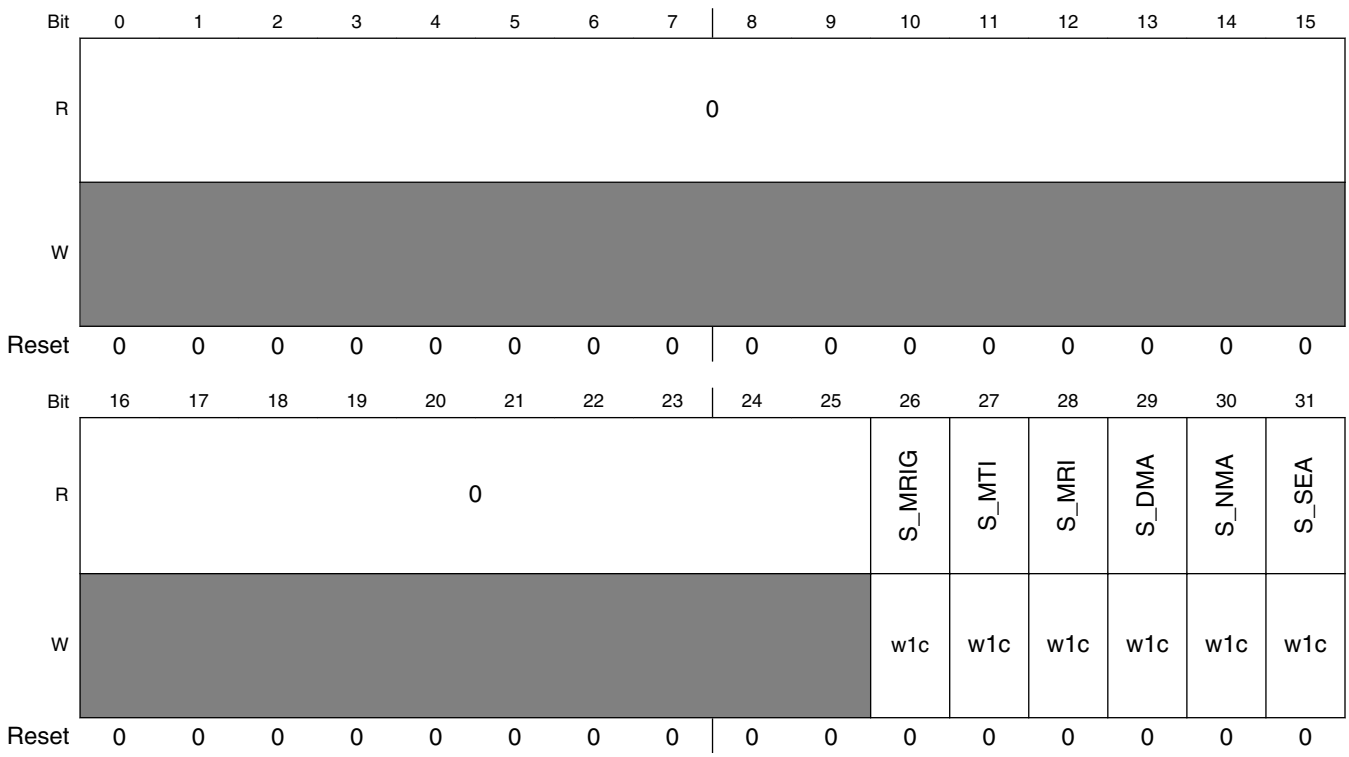
MC_ME_IM field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 M_ICONF_CC	Invalid mode configuration (core configuration) interrupt mask 0 Invalid mode configuration (core configuration) interrupt is masked 1 Invalid mode configuration (core configuration) interrupt is enabled
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 M_ICONF	Invalid mode configuration interrupt mask 0 Invalid mode configuration (core configuration) interrupt is masked 1 Invalid mode configuration (core configuration) interrupt is enabled
29 M_IMODE	Invalid mode interrupt mask 0 Invalid mode interrupt is masked 1 Invalid mode interrupt is enabled
30 M_SAFE	SAFE mode interrupt mask 0 SAFE mode interrupt is masked 1 SAFE mode interrupt is enabled
31 M_MTC	Mode transition complete interrupt mask 0 Mode transition complete interrupt is masked 1 Mode transition complete interrupt is enabled

40.3.6 Invalid Mode Transition Status Register (MC_ME_IMTS)

This register provides the status bits for the possible causes of an invalid mode interrupt.

Address: 0h base + 14h offset = 14h



MC_ME_IMTS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 S_MRIG	Mode Request Ignored status — This bit is set whenever a new mode is requested while a transition to the SAFE mode is in progress. It is cleared by writing a ‘1’ to this bit. This bit is also set when a transition to STANDBY0/STOP0 mode is requested while a wakeup event is active. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal
27 S_MTI	Mode Transition Illegal status — This bit is set whenever a new mode is requested while some other non-SAFE mode transition process is active (S_MTRANS is ‘1’). Please refer to Mode Transition Interrupts for the exceptions to this behavior. It is cleared by writing a ‘1’ to this bit. 0 Mode transition requested is not illegal 1 Mode transition requested is illegal

Table continues on the next page...

MC_ME_IMTS field descriptions (continued)

Field	Description
28 S_MRI	Mode Request Illegal status — This bit is set whenever the target mode requested is not a valid mode with respect to current mode. It is cleared by writing a ‘1’ to this bit. 0 Target mode requested is not illegal with respect to current mode 1 Target mode requested is illegal with respect to current mode
29 S_DMA	Disabled Mode Access status — This bit is set whenever the target mode requested is one of those disabled modes determined by ME_ME register. It is cleared by writing a ‘1’ to this bit. 0 Target mode requested is not a disabled mode 1 Target mode requested is a disabled mode
30 S_NMA	Non-existing Mode Access status — This bit is set whenever the target mode requested is one of those non existing modes determined by ME_ME register. It is cleared by writing a ‘1’ to this bit. 0 Target mode requested is an existing mode 1 Target mode requested is a non-existing mode
31 S_SEA	SAFE Event Active status — This bit is set whenever the chip is in SAFE mode, SAFE event bit is pending and a new mode requested other than RESET/SAFE modes. It is cleared by writing a ‘1’ to this bit. 0 No new mode requested other than RESET/SAFE while SAFE event is pending 1 New mode requested other than RESET/SAFE while SAFE event is pending

40.3.7 Debug Mode Transition Status Register (MC_ME_DMTS)

This register provides the status of different factors which influence mode transitions. It is used to give an indication of why a mode transition indicated by ME_GS.S_MTRANS may be taking longer than expected. NOTE The ME_DMTS register does not indicate whether a mode transition is ongoing. Therefore, some ME_DMTS bits may still be asserted after the mode transition has completed.

NOTE

After BAF exits, and the control moves to application code, DMTS register is not in Default Reset State. It shows 0x30000000 which implies previous mode was DRUN.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PREVIOUS_MODE				0				MPH_BUSY	0		PMC_PROG	DBG_MODE	CCKL_PROG	PCS_PROG	SMR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VREG_CSRC_SC	CSRC_CSRC_SC	IRC_SC	SCSRC_SC	SYSCLK_SW	0	FLASH_SC	0				CDP_PRPH_96_127	CDP_PRPH_64_95	CDP_PRPH_32_63	CDP_PRPH_0_31
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_DMTS field descriptions

Field	Description
0–3 PREVIOUS_ MODE	Previous chip mode — These bits show the mode in which the chip was prior to the latest change to the current mode. **NOTE:** Mode codes “1001”, “1011”, “1100”, “1110”, and “1111” are reserved and should never appear. 0000 RESET 0001 Reserved 0010 SAFE 0011 DRUN 0100 RUN0 0101 RUN1 0110 RUN2 0111 RUN3 1000 Reserved 1001 reserved 1010 STOP0 1011 reserved 1100 reserved 1101 STANDBY0[1] 1110 reserved 1111 reserved
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_DMTS field descriptions (continued)

Field	Description
8 MPH_BUSY	MC_ME/MC_PCU Handshake Busy indicator — This bit is set if the MC_ME has requested a mode change from the MC_PCU and the MC_PCU has not yet completed its power-up/down sequencing. It is cleared when the MC_PCU has power-up/down sequencing. 0 Handshake is not busy 1 Handshake is busy
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 PMC_PROG	MC_PCU Mode Change in Progress indicator — This bit is set if the MC_PCU is in the process of powering up or down power domains. It is cleared when all power-up/down processes have completed. 0 Power-up/down transition is not in progress 1 Power-up/down transition is in progress
12 DBG_MODE	Debug mode indicator — This bit is set while the chip is in debug mode. 0 The chip is not in debug mode 1 The chip is in debug mode
13 CCKL_PROG	Core Clock Enable/Disable in Progress — This bit is set while any core's clock is in the process of being enabled or disabled. 0 No core clock is being enabled or disabled 1 A core clock is being enabled or disabled
14 PCS_PROG	Progressive System Clock Switching in Progress — This bit is set while the progressive system clock switching process is in progress. The PCS_PROG bit is set when the • PCS is in progress or • Peripheral clock switching is ongoing or • Core clock switching is ongoing or • Power sequence process is ongoing (provided the target mode is not STANDBY) 0 PCS is not in progress 1 PCS is in progress
15 SMR	SAFE mode request from MC_RGM is active indicator — This bit is set if a hardware SAFE mode request has been triggered. It is cleared when the hardware SAFE mode request has been cleared. 0 A SAFE mode request is not active 1 A SAFE mode request is active
16 Reserved	Clock Disable Process Pending status for Peripherals 0…255 This read-only field is reserved and always has the value 0. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
17 VREG_CSRC_SC	Main VREG dependent Clock Source State Change during mode transition indicator — This bit is set when a clock source which depends on the main voltage regulator to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place

Table continues on the next page...

MC_ME_DMTS field descriptions (continued)

Field	Description
18 CSRC_CSRC_SC	(Other) Clock Source dependent Clock Source State Change during mode transition indicator — This bit is set when a clock source which depends on another clock source to be powered-up is requested to change its power up/down state. It is cleared when the clock source has completed its state change. 0 No state change is taking place 1 A state change is taking place
19 IRC_SC	IRC State Change during mode transition indicator — This bit is set when the 16 MHz internal RC oscillator is requested to change its power up/down state. It is cleared when the 16 MHz internal RC oscillator has completed its state change. 0 No state change is taking place 1 A state change is taking place
20 SCSRC_SC	Secondary Clock Sources State Change during mode transition indicator — This bit is set when a secondary clock source is requested to change its power up/down state. It is cleared when all secondary system clock sources have completed their state changes. (A 'secondary clock source' is a clock source other than IRC.) 0 No state change is taking place 1 A state change is taking place
21 SYSCLK_SW	System Clock Switching pending status — 0 No system clock source switching is pending 1 A system clock source switching is pending
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FLASH_SC	FLASH State Change during mode transition indicator — This bit is set when the FLASH is requested to change its power up/down state. It is cleared when the DFLASH has completed its state change. 0 No state change is taking place 1 A state change is taking place
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 CDP_PRPH_96_127	Clock Disable Process Pending status for Peripherals 96…127 — This bit is set when any peripheral appearing in ME_PS3 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. Peripheral n corresponds to the ME_PCTLn register. Please refer to memeory map table for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
29 CDP_PRPH_64_95	Clock Disable Process Pending status for Peripherals 64…95 — This bit is set when any peripheral appearing in ME_PS2 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

Table continues on the next page...

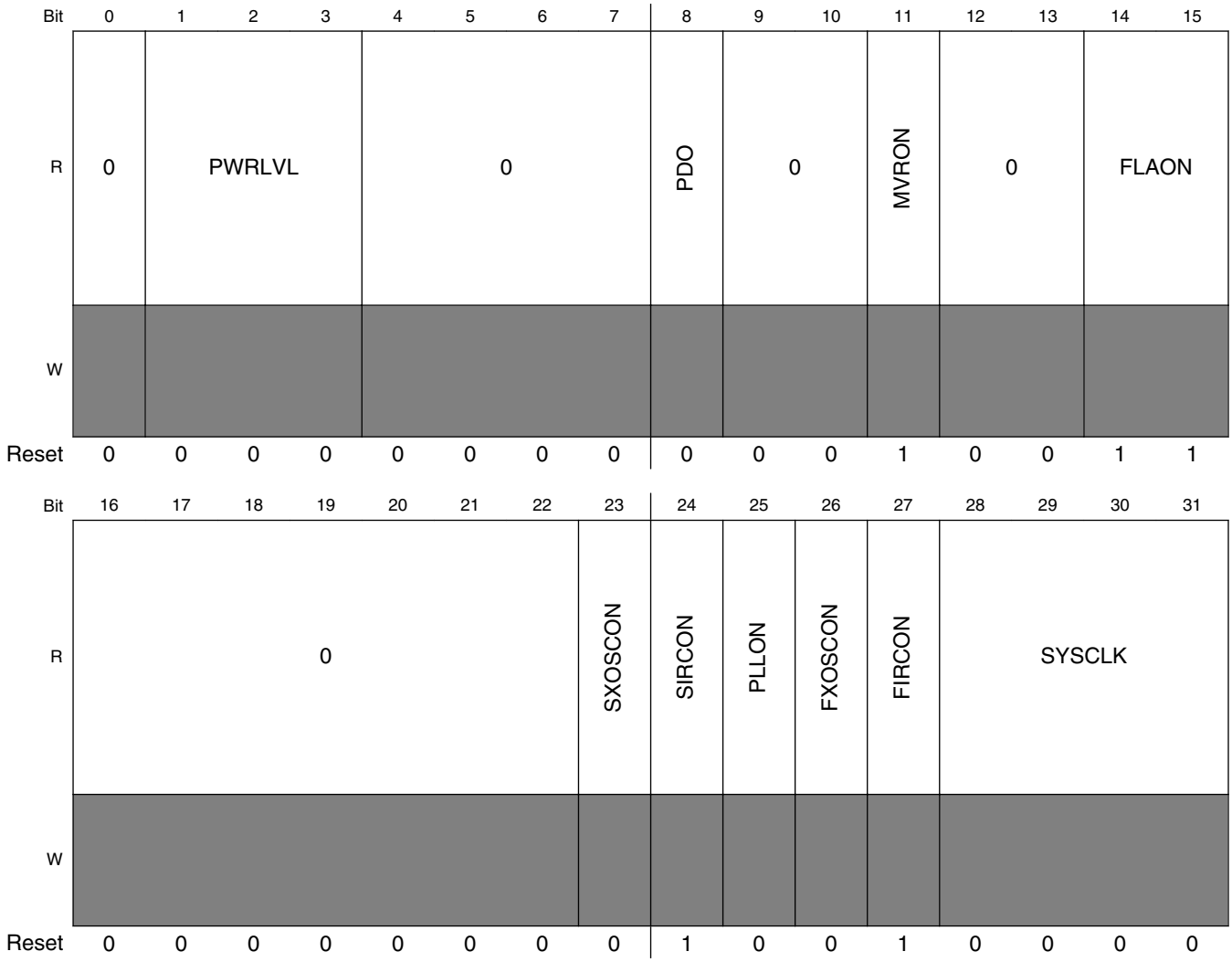
MC_ME_DMTS field descriptions (continued)

Field	Description
30 CDP_PRPH_32_63	Clock Disable Process Pending status for Peripherals 32…63 — This bit is set when any peripheral appearing in ME_PS1 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. Peripheral n corresponds to the ME_PCTLn register. Please refer to memeory map table for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral
31 CDP_PRPH_0_31	Clock Disable Process Pending status for Peripherals 0…31 — This bit is set when any peripheral appearing in ME_PS0 has been requested to have its clock disabled. It is cleared when all these peripherals which have been requested to have their clocks disabled have entered the state in which their clocks may be disabled. Peripheral n corresponds to the ME_PCTLn register. Please refer to memeory map table for the ME_PCTLn locations actually occupied, which in turn indicates which peripherals are reported in the ME_DMTS register. 0 No peripheral clock disabling is pending 1 Clock disabling is pending for at least one peripheral

1. This register/bitfield does not give a valid indication for the previous_mode after wakeup from STANDBY because MC_ME registers are power gated during STANDBY mode. See PMCDIG_SGSR[LP_STATUS] for STANDBY status.

40.3.8 RESET Mode Configuration Register (MC_ME_RESET_MC)

Address: 0h base + 20h offset = 20h



MC_ME_RESET_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os.

Table continues on the next page...

MC_ME_RESET_MC field descriptions (continued)

Field	Description
	0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE mode, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 reserved 11 Flash is in RUN mode
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SXOSCON	32Khz external oscillator control 0 32Khz external oscillator control is switched off 1 32Khz external oscillator control is switched on.
24 SIRCON	128Khz internal RC oscillator control 0 128Khz internal RC oscillator control is switched off 1 128Khz internal RC oscillator control is switched on
25 PLLON	PLL status 0 PLL is switched off 1 PLL is switched on
26 FXOSCON	8-40 MHz crystal oscillator control 0 8-40 MHz crystal oscillator is switched off 1 8-40 MHz crystal oscillator is switched on
27 FIRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 8-40 MHz crystal osc. 0010 PLL PHI_0 0011 reserved 0100 reserved 0101 reserved

Table continues on the next page...

MC_ME_RESET_MC field descriptions (continued)

Field	Description
	0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 Reserved

40.3.9 SAFE Mode Configuration Register (MC_ME_SAFE_MC)

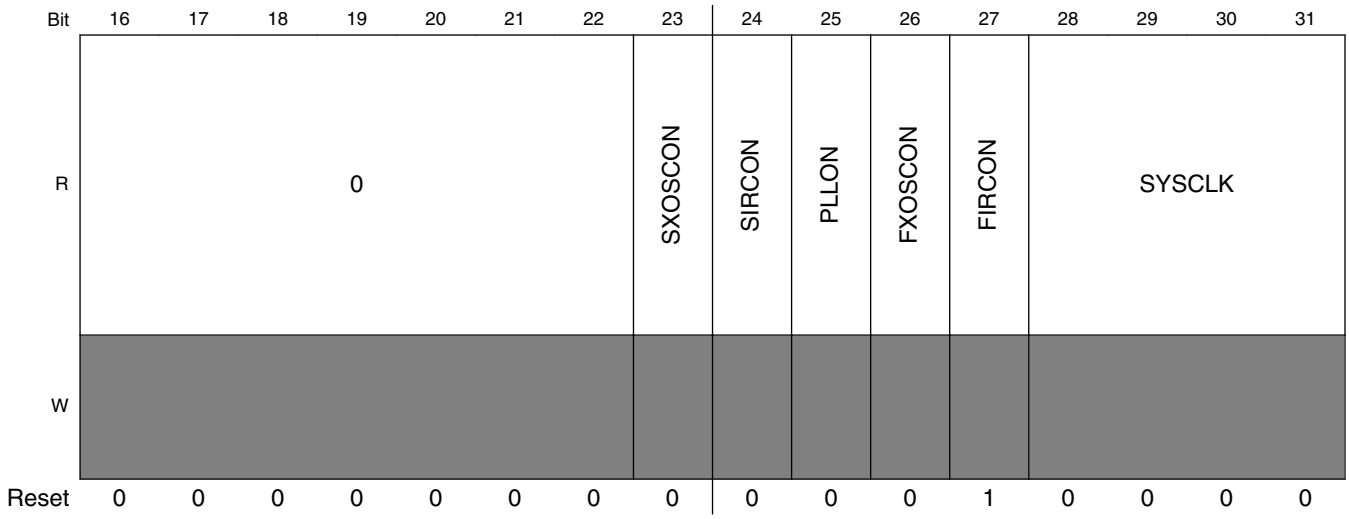
This register configures system behavior during SAFE mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1



MC_ME_SAFE_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE mode, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 reserved 11 Flash is in RUN mode
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SXOSCON	32Khz external oscillator control

Table continues on the next page...

MC_ME_SAFE_MC field descriptions (continued)

Field	Description
	0 32Khz external oscillator control 1 32Khz external oscillator control
24 SIRCON	128Khz internal RC oscillator control 0 128Khz internal RC oscillator control is switched off 1 128Khz internal RC oscillator control is switched on
25 PLLON	PLL control 0 PLL is switched off 1 PLL is switched on
26 FXOSCON	8-40 MHz crystal oscillator control 0 8-40 MHz crystal oscillator is switched off 1 8-40 MHz crystal oscillator is switched on
27 FIRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 Reserved 0010 Reserved 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

40.3.10 DRUN Mode Configuration Register (MC_ME_DRUN_MC)

This register configures system behavior during DRUN mode.

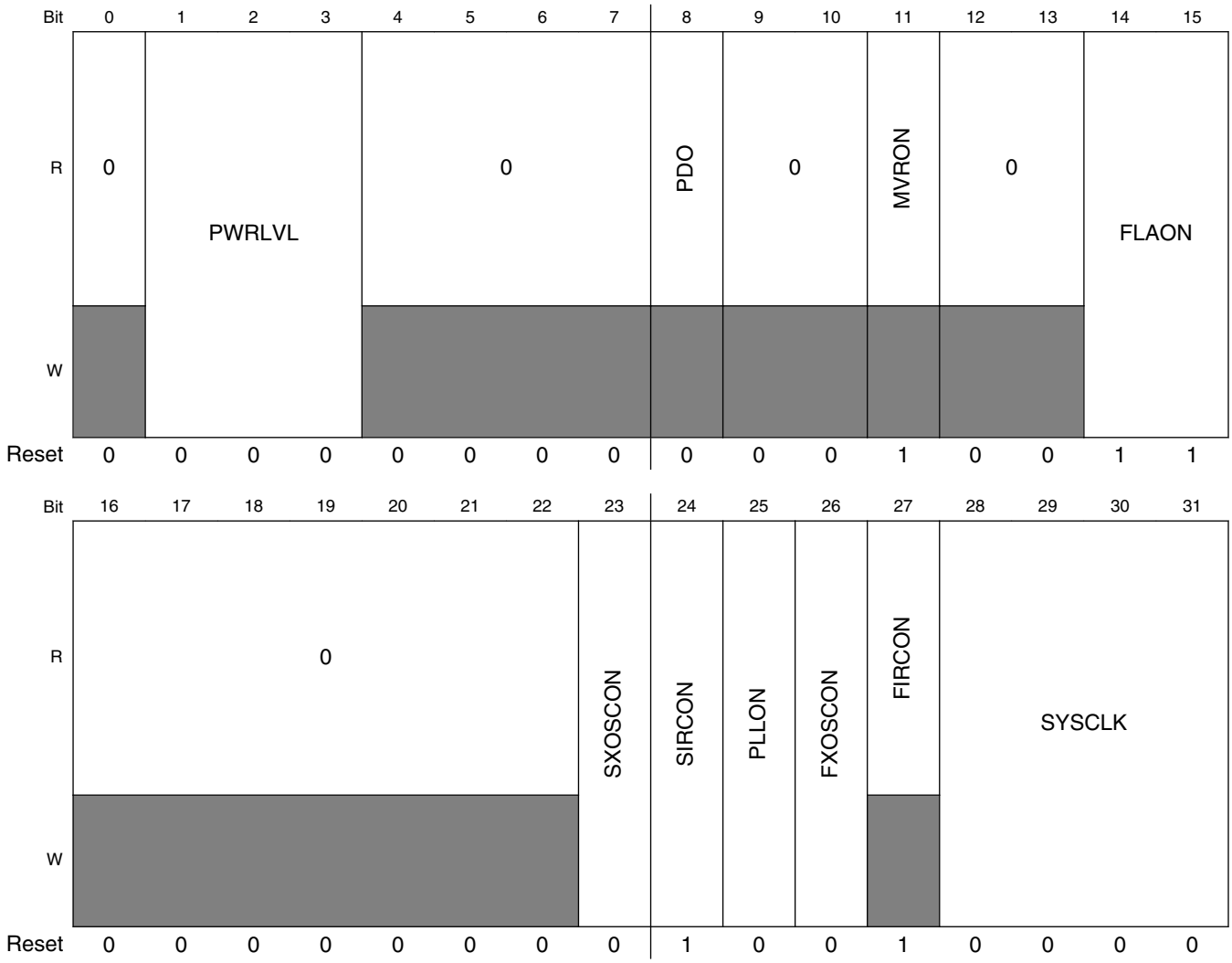
NOTE

Byte write accesses are not allowed to this register.

NOTE

The following configuration values are set according to the chip configuration: FXOSCON

Address: 0h base + 2Ch offset = 2Ch



MC_ME_DRUN_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os.

Table continues on the next page...

MC_ME_DRUN_MC field descriptions (continued)

Field	Description
	0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE mode, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 reserved 11 Flash is in RUN mode
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SXOSCON	32Khz external oscillator control 0 32Khz external oscillator control 1 32Khz external oscillator control
24 SIRCON	128 KHz internal RC oscillator control 0 128 KHz internal RC oscillator control is switched off 1 128 KHz internal RC oscillator control is switched on
25 PLLON	PLL control 0 PLL is switched off 1 PLL is switched on
26 FXOSCON	8-40 MHz crystal oscillator control 0 8-40 MHz crystal oscillator is switched off 1 8-40 MHz crystal oscillator is switched on
27 FIRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 8-40 MHz crystal osc. 0010 PLL PHI_0 0011 reserved 0100 reserved 0101 reserved

Table continues on the next page...

MC_ME_DRUN_MC field descriptions (continued)

Field	Description
	0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 Reserved

40.3.11 RUN0 Mode Configuration Register (MC_ME_RUN0_MC)

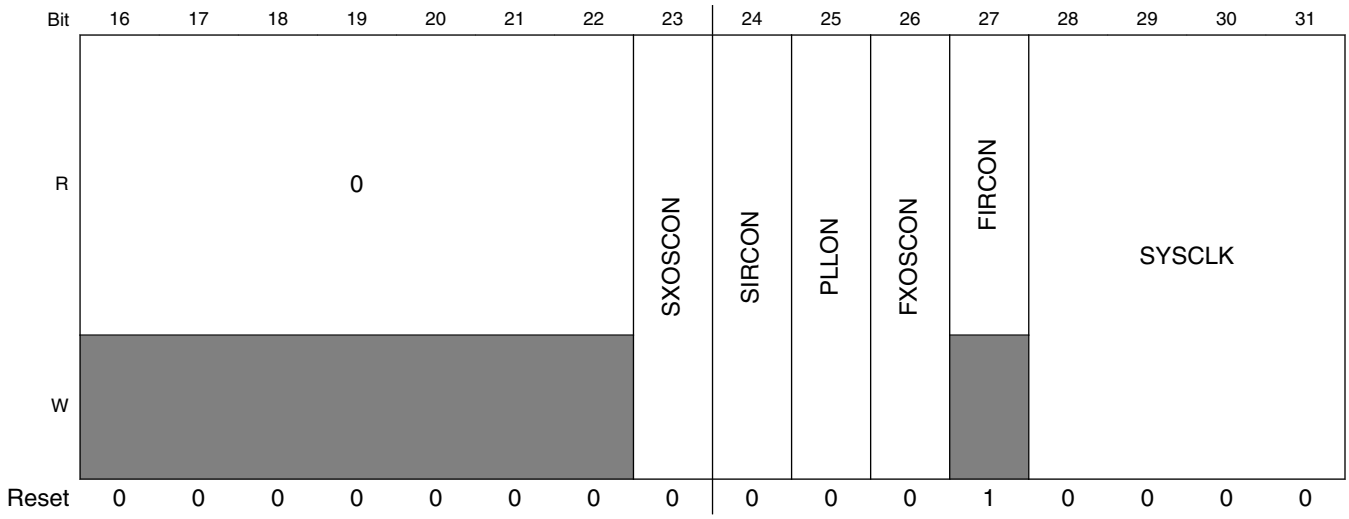
This register configures system behavior during RUN0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1



MC_ME_RUN0_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE mode, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 reserved 11 Flash is in RUN mode
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SXOSCON	32 KHz external oscillator control

Table continues on the next page...

MC_ME_RUN0_MC field descriptions (continued)

Field	Description
	0 32Khz external oscillator control is off 1 32Khz external oscillator control is on
24 SIRCON	128 KHz internal RC oscillator control 0 128 KHz internal RC oscillator control is switched off 1 128 KHz internal RC oscillator control is switched on
25 PLLON	PLL control 0 PLL is switched off 1 PLL is switched on
26 FXOSCON	8-40 MHz crystal oscillator control 0 8-40 MHz crystal oscillator is switched off 1 8-40 MHz crystal oscillator is switched on
27 FIRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 8-40 MHz crystal osc. 0010 PLL PHI_0 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 Reserved

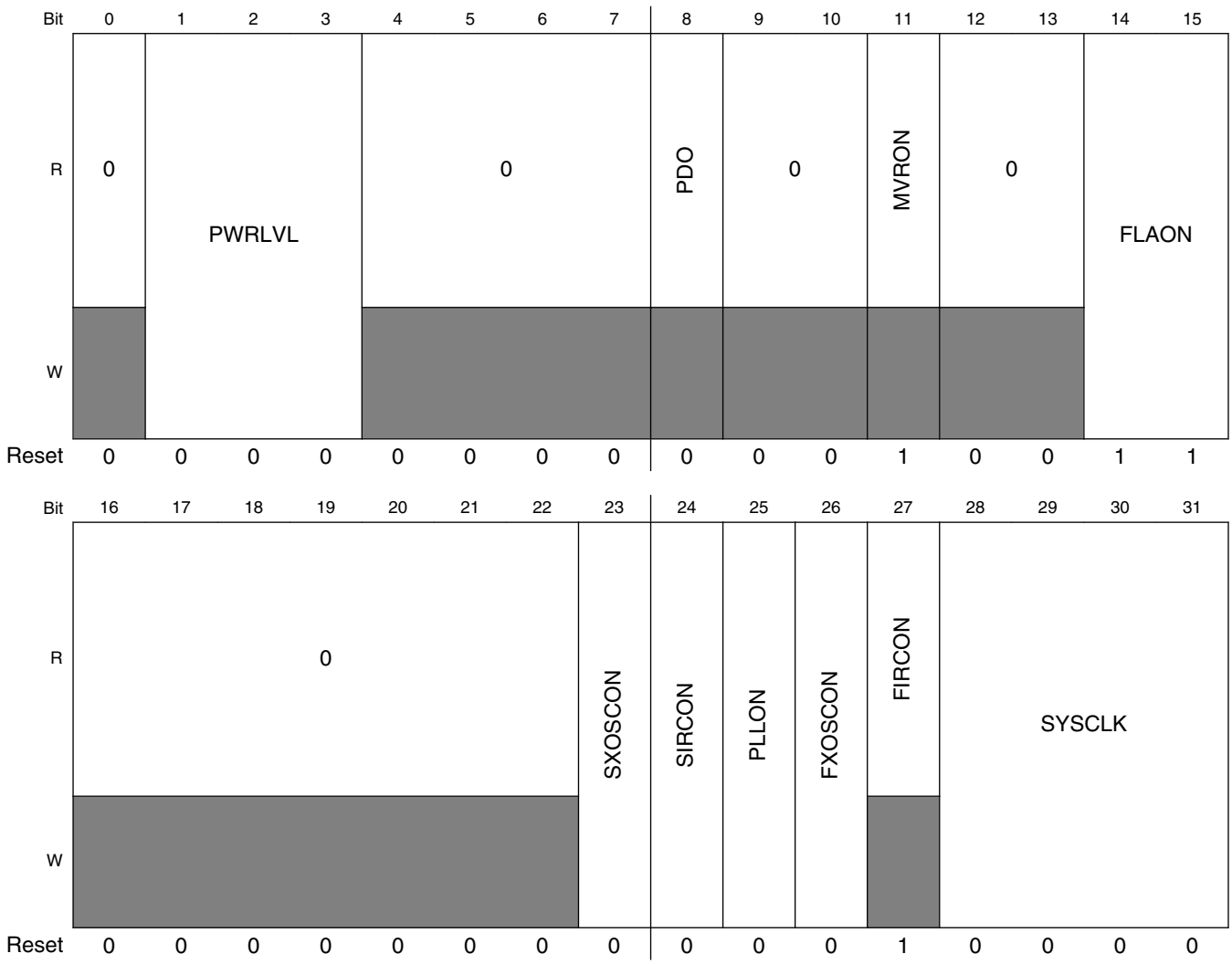
40.3.12 RUN1 Mode Configuration Register (MC_ME_RUN1_MC)

This register configures system behavior during RUN1 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 34h offset = 34h



MC_ME_RUN1_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE mode, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_RUN1_MC field descriptions (continued)

Field	Description
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 reserved 11 Flash is in RUN mode
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SXOSCON	32 KHz external oscillator control 0 32 KHz external oscillator control is off 1 32 KHz external oscillator control is on
24 SIRCON	128 KHz internal RC oscillator control 0 128 KHz internal RC oscillator control is switched off 1 128 KHz internal RC oscillator control is switched on
25 PLLON	PLL control 0 PLL is switched off 1 PLL is switched on
26 FXOSCON	8-40 MHz crystal oscillator control 0 8-40 MHz crystal oscillator is switched off 1 8-40 MHz crystal oscillator is switched on
27 FIRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 8-40 MHz crystal osc. 0010 PLL PHI_0 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved

Table continues on the next page...

MC_ME_RUN1_MC field descriptions (continued)

Field	Description
	1100 reserved 1101 reserved 1110 reserved 1111 Reserved

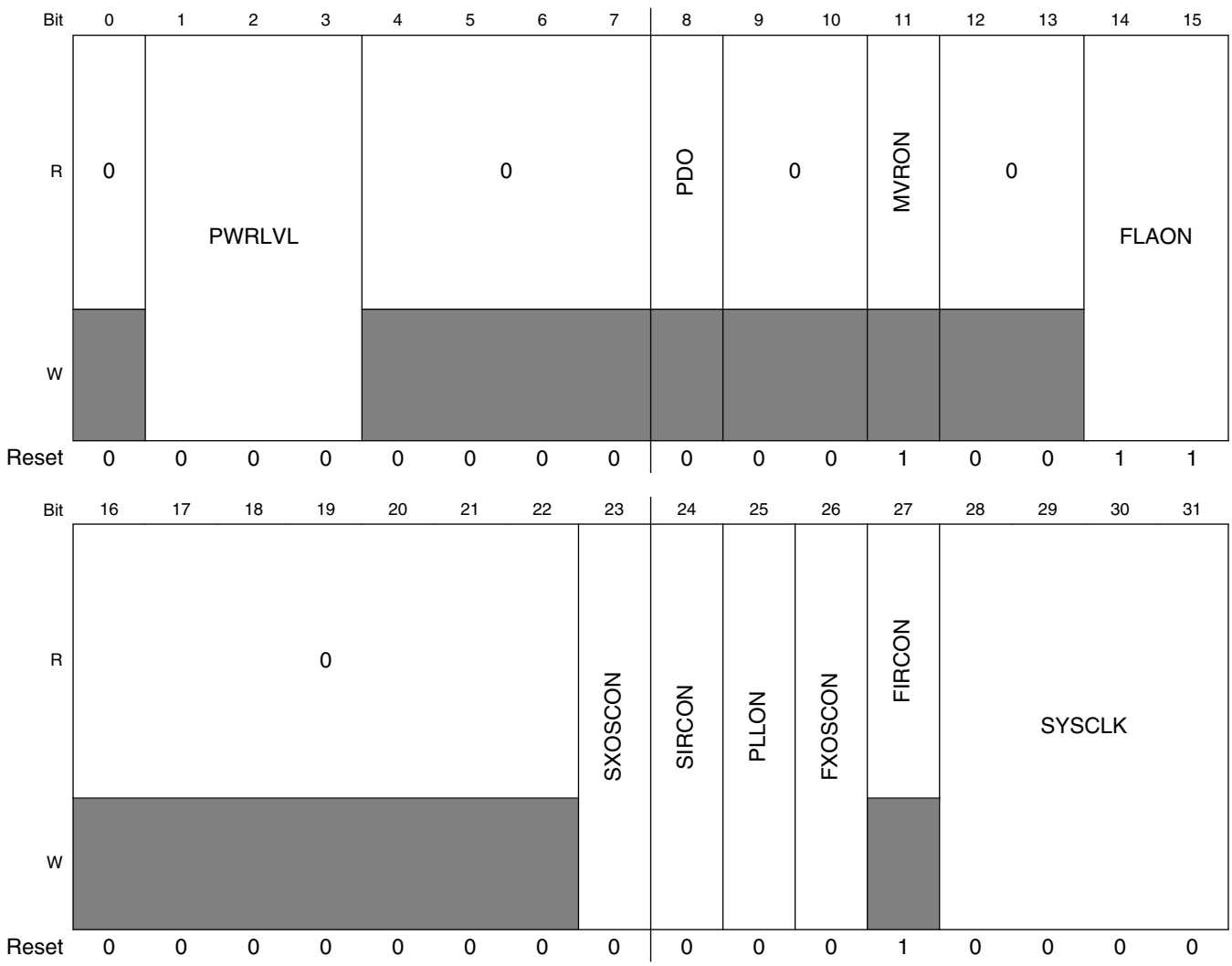
40.3.13 RUN2 Mode Configuration Register (MC_ME_RUN2_MC)

This register configures system behavior during RUN1 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 38h offset = 38h



MC_ME_RUN2_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE mode, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 reserved 11 Flash is in RUN mode
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SXOSCON	32 KHz external oscillator control 0 32 KHz external oscillator control is off 1 32 KHz external oscillator control is on
24 SIRCON	128 KHz internal RC oscillator control 0 128 KHz internal RC oscillator control is switched off 1 128 KHz internal RC oscillator control is switched on
25 PLLON	PLL control 0 PLL is switched off 1 PLL is switched on
26 FXOSCON	8-40 MHz crystal oscillator control 0 8-40 MHz crystal oscillator is switched off 1 8-40 MHz crystal oscillator is switched on
27 FIRCON	16 MHz internal RC oscillator control

Table continues on the next page...

MC_ME_RUN2_MC field descriptions (continued)

Field	Description
	0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 8-40 MHz crystal osc. 0010 PLL PHI_0 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 Reserved

40.3.14 RUN3 Mode Configuration Register (MC_ME_RUN3_MC)

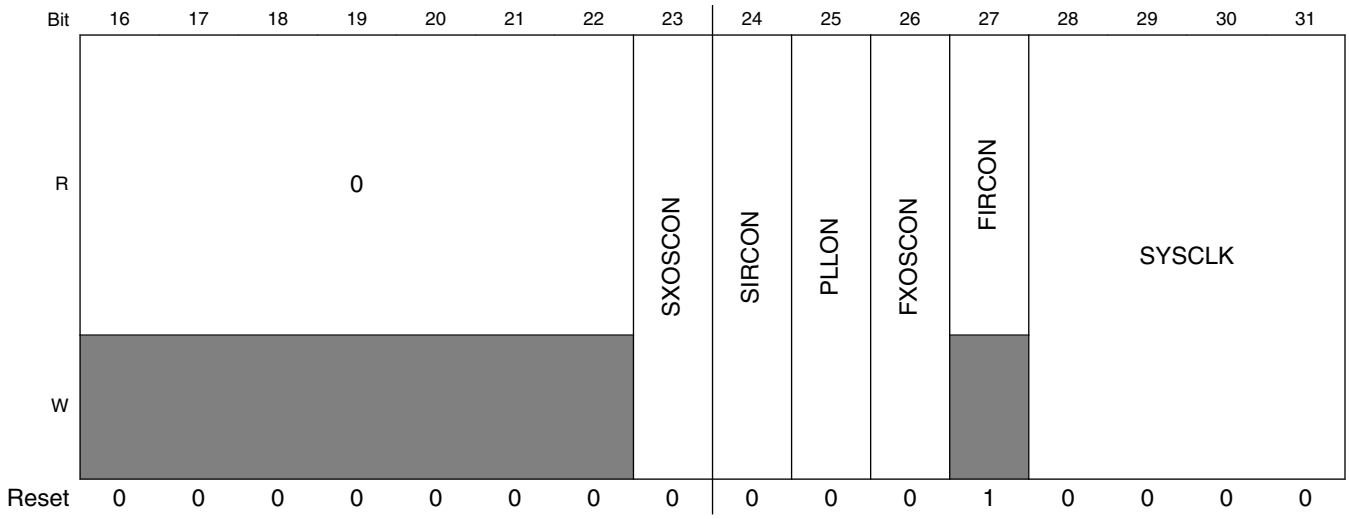
This register configures system behavior during RUN3 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1



MC_ME_RUN3_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE mode, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 reserved 11 Flash is in RUN mode
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SXOSCON	32 KHz external oscillator control

Table continues on the next page...

MC_ME_RUN3_MC field descriptions (continued)

Field	Description
	0 32 KHz external oscillator control is off 1 32 KHz external oscillator control in off
24 SIRCON	128 KHz internal RC oscillator control 0 128 KHz internal RC oscillator control is switched off 1 128 KHz internal RC oscillator control is switched on
25 PLLON	PLL control 0 PLL is switched off 1 PLL is switched on
26 FXOSCON	8-40 MHz crystal oscillator control 0 8-40 MHz crystal oscillator is switched off 1 8-40 MHz crystal oscillator is switched on
27 FIRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 8-40 MHz crystal osc. 0010 PLL PHI_0 0011 reserved 0100 reserved 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 Reserved

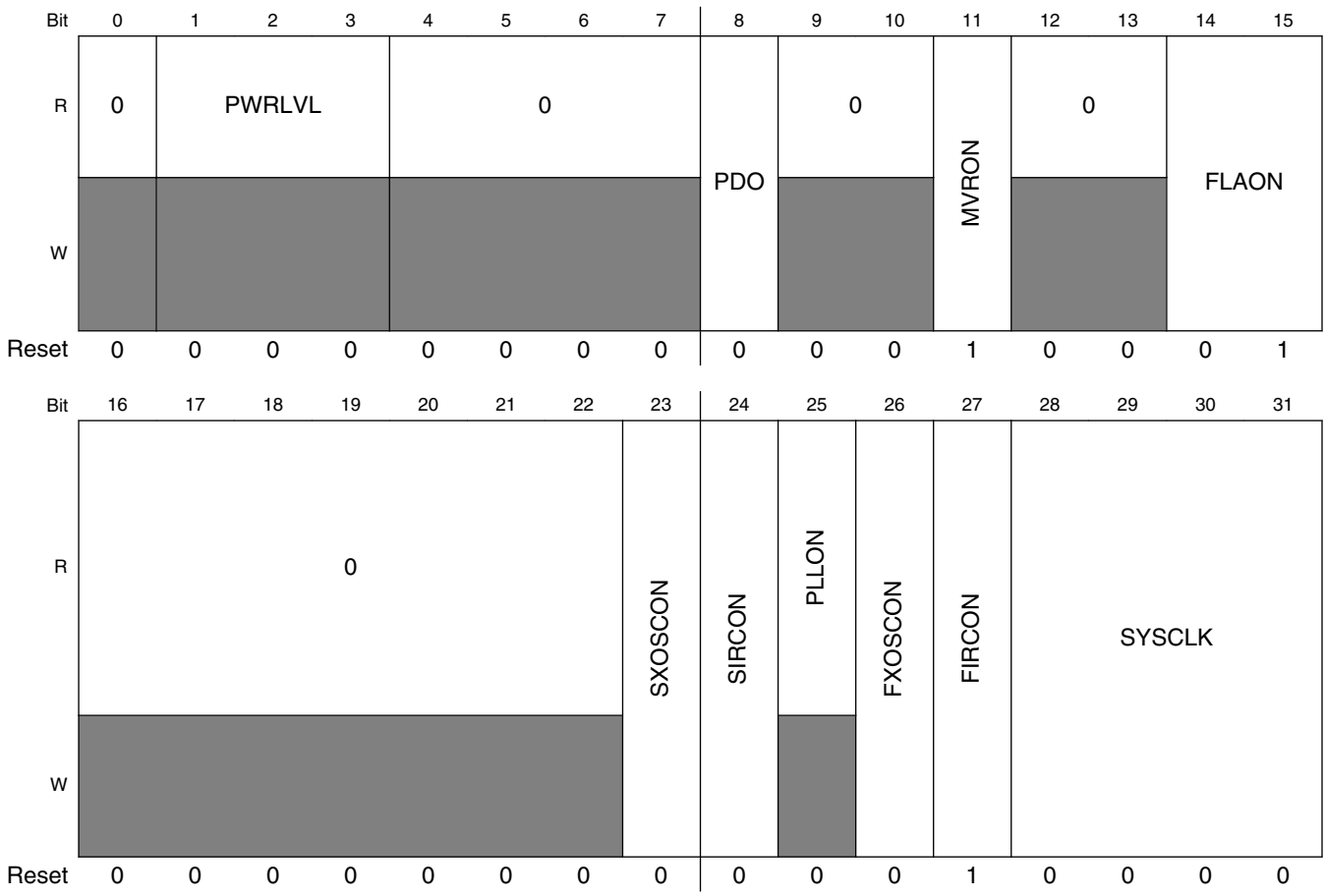
40.3.15 STOP0 Mode Configuration Register (MC_ME_STOP_MC)

This register configures system behavior during STOP0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 48h offset = 48h



MC_ME_STOP_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. **NOTE:** Power-sequence drivers control pad slopes during run mode and are disabled in certain System or User modes to further reduce power consumption. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE mode, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode.

Table continues on the next page...

MC_ME_STOP_MC field descriptions (continued)

Field	Description
	NOTE: It is controlled in hardware that, accidentally main voltage regulator is not turned OFF. 1 Main voltage regulator is switched on 0 No impact
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 reserved 11 Flash is in RUN mode
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SXOSCON	32 KHz external oscillator control 0 32 KHz external oscillator control is off 1 32 KHz external oscillator control is on
24 SIRCON	128 KHz internal RC oscillator control 0 128 KHz internal RC oscillator control is switched off 1 128 KHz internal RC oscillator control is switched on
25 PLLON	PLL control 0 PLL is switched off 1 PLL is switched on
26 FXOSCON	8-40 MHz crystal oscillator control 0 8-40 MHz crystal oscillator is switched off 1 8-40 MHz crystal oscillator is switched on
27 FIRCON	16 MHz internal RC oscillator control 0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 16 MHz int. RC osc. 0001 8-40 MHz crystal osc. 0010 Reserved 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved

Table continues on the next page...

MC_ME_STOP_MC field descriptions (continued)

Field	Description
	1100 Reserved 1101 Reserved 1110 Reserved 1111 No System Clock available

40.3.16 STANDBY0 Mode Configuration Register (MC_ME_STANDBY_MC)

This register configures system behavior during STANDBY0 mode.

NOTE

Byte write accesses are not allowed to this register.

Address: 0h base + 54h offset = 54h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PWRLVL			0				PDO	0		MVRON	0		FLAON	
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0							SXOSCON	SIRCON	PLLON	FXOSCON	FIRCON	SYSCLK			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

MC_ME_STANDBY_MC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–3 PWRLVL	Power level — These bits indicate the relative power consumption level of this mode with respect to that of other modes. When switching between two modes with differing power levels, system clock progressive switching is enabled.
4–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 PDO	I/O output power-down control — This bit controls the output power-down of I/Os. 0 No automatic safe gating of I/Os used and pads power sequence driver is enabled 1 In SAFE, outputs of pads are forced to high impedance state and pads power sequence driver is disabled. The inputs are level unchanged. In STOP0 mode, only the pad power sequence driver is disabled, but the state of the output remains functional.
9–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 MVRON	Main voltage regulator control — This bit specifies whether main voltage regulator is switched off or not while entering this mode. 1 Main voltage regulator is switched on 0 Main voltage regulator is switched off
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 FLAON	Flash power-down control — This bit specifies the operating mode of the code flash after entering this mode. 00 reserved 01 Flash is in power-down mode 10 reserved 11 Flash is in RUN mode
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SXOSCON	32 KHz external oscillator control 0 32 KHz external oscillator control is off 1 32 KHz external oscillator control is on
24 SIRCON	128 KHz internal RC oscillator control 0 128 KHz internal RC oscillator control is switched off 1 128 KHz internal RC oscillator control is switched on
25 PLLON	PLL control 0 PLL is switched off 1 PLL is switched on
26 FXOSCON	8-40 MHz crystal oscillator control 0 8-40 MHz crystal oscillator is switched off 1 8-40 MHz crystal oscillator is switched on
27 FIRCON	16 MHz internal RC oscillator control

Table continues on the next page...

MC_ME_STANDBY_MC field descriptions (continued)

Field	Description
	0 16 MHz internal RC oscillator is switched off 1 16 MHz internal RC oscillator is switched on
28–31 SYSCLK	System clock switch control — These bits specify the system clock to be used by the system. 0000 Reserved 0001 Reserved 0010 Reserved 0011 Reserved 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 No System Clock available

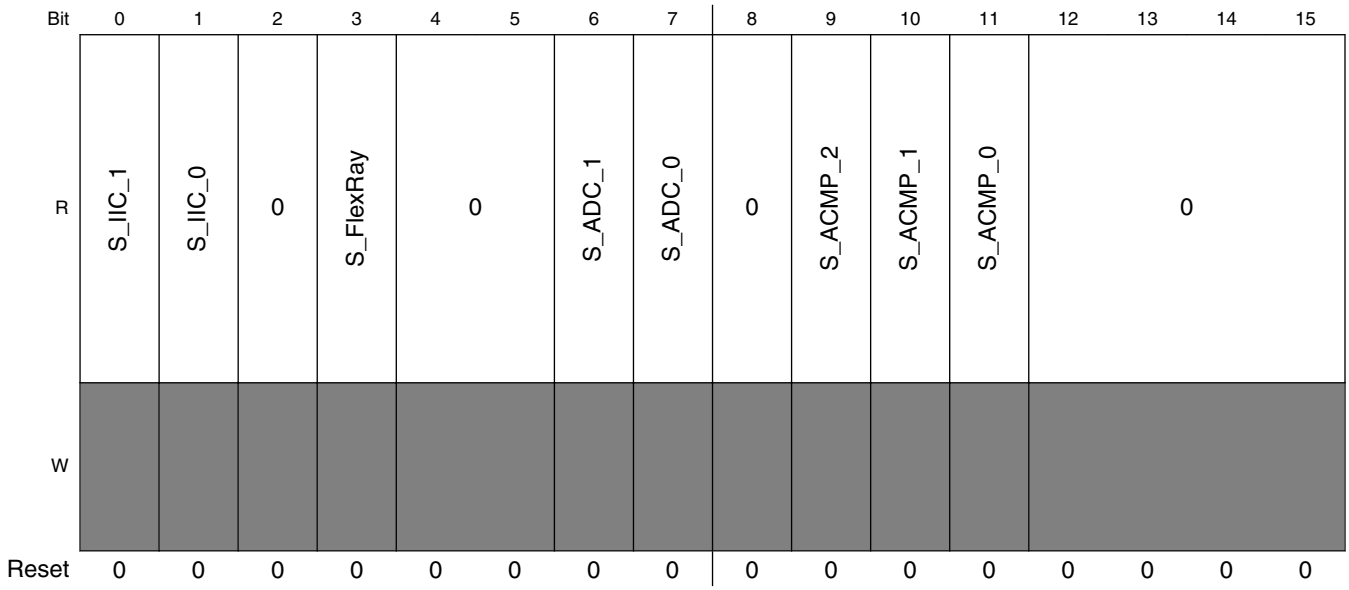
40.3.17 Peripheral Status Register 0 (MC_ME_PS0)

This register provides the status of the peripherals.

NOTE

Peripheral Status for ENET and PIT will be 1 when both ME_PCTL is enabled and the clock gating control bits within IP are also enabled. For other modules, Peripheral Status will depend on direct control through ME_PCTL.

Address: 0h base + 60h offset = 60h



MC_ME_PS0 field descriptions

Field	Description
0 S_IIC_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
1 S_IIC_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active

Table continues on the next page...

MC_ME_PS0 field descriptions (continued)

Field	Description
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 S_FlexRay	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 S_ADC_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active .
7 S_ADC_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 S_ACMP_2	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
10 S_ACMP_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
11 S_ACMP_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 S_ENET	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
17–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 S_SAI_2	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen

Table continues on the next page...

MC_ME_PS0 field descriptions (continued)

Field	Description
	1: Peripheral is active
20 S_SAI_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active .
21 S_SAI_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
22–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 S_eMIOS_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME.
30 S_eMIOS_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME.
31 S_BCTU	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active

40.3.18 Peripheral Status Register 1 (MC_ME_PS1)

This register provides the status of the peripherals.

Address: 0h base + 64h offset = 64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	S_LIN[13:0]														0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				S_DSPI[3:0]				0						S_IIC[3:2]	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

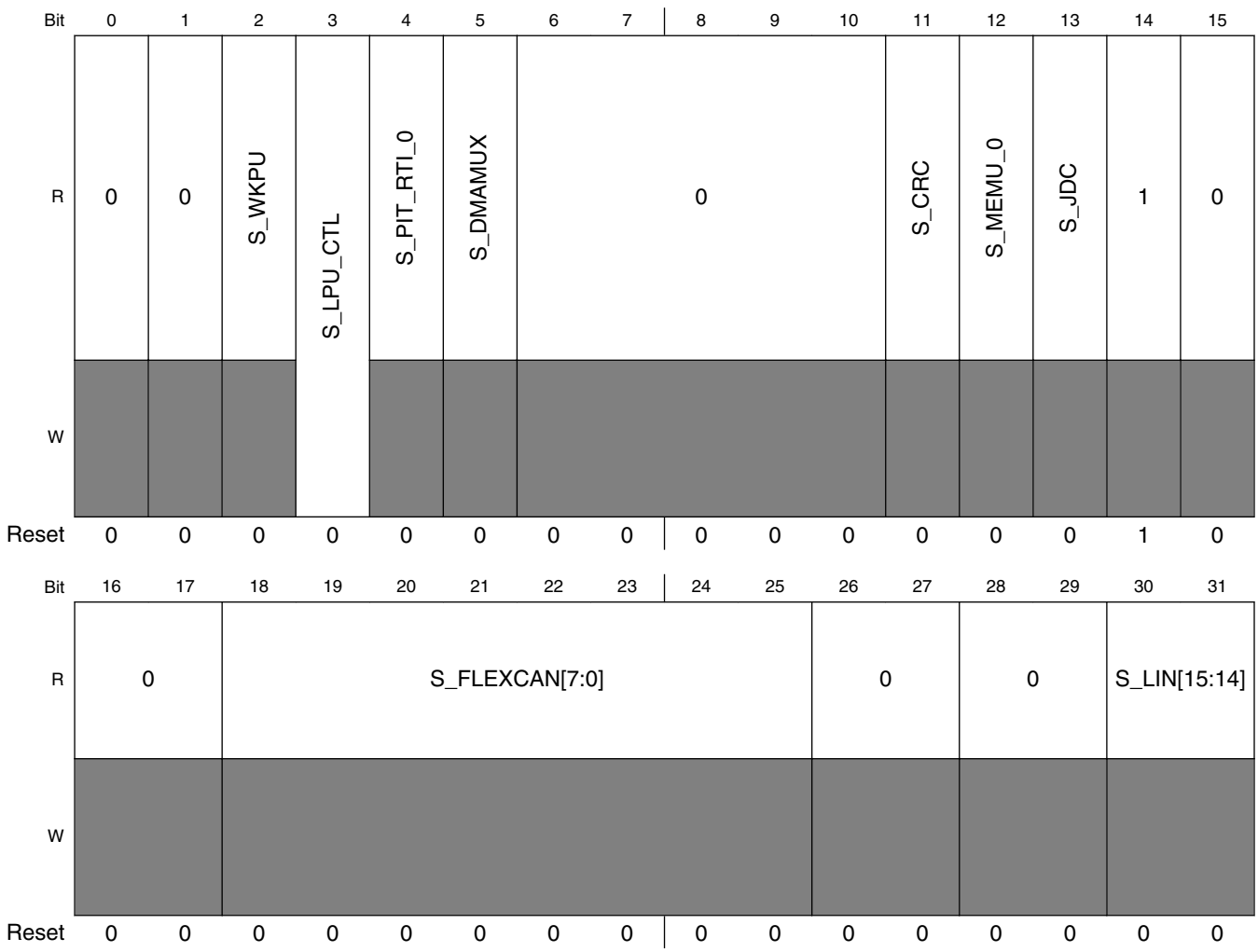
MC_ME_PS1 field descriptions

Field	Description
0–13 S_LIN[13:0]	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
14–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 S_DSPI[3:0]	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
24–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 S_IIC[3:2]	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active

40.3.19 Peripheral Status Register 2 (MC_ME_PS2)

This register provides the status of the peripherals.

Address: 0h base + 68h offset = 68h



MC_ME_PS2 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 S_WKPU	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen

Table continues on the next page...

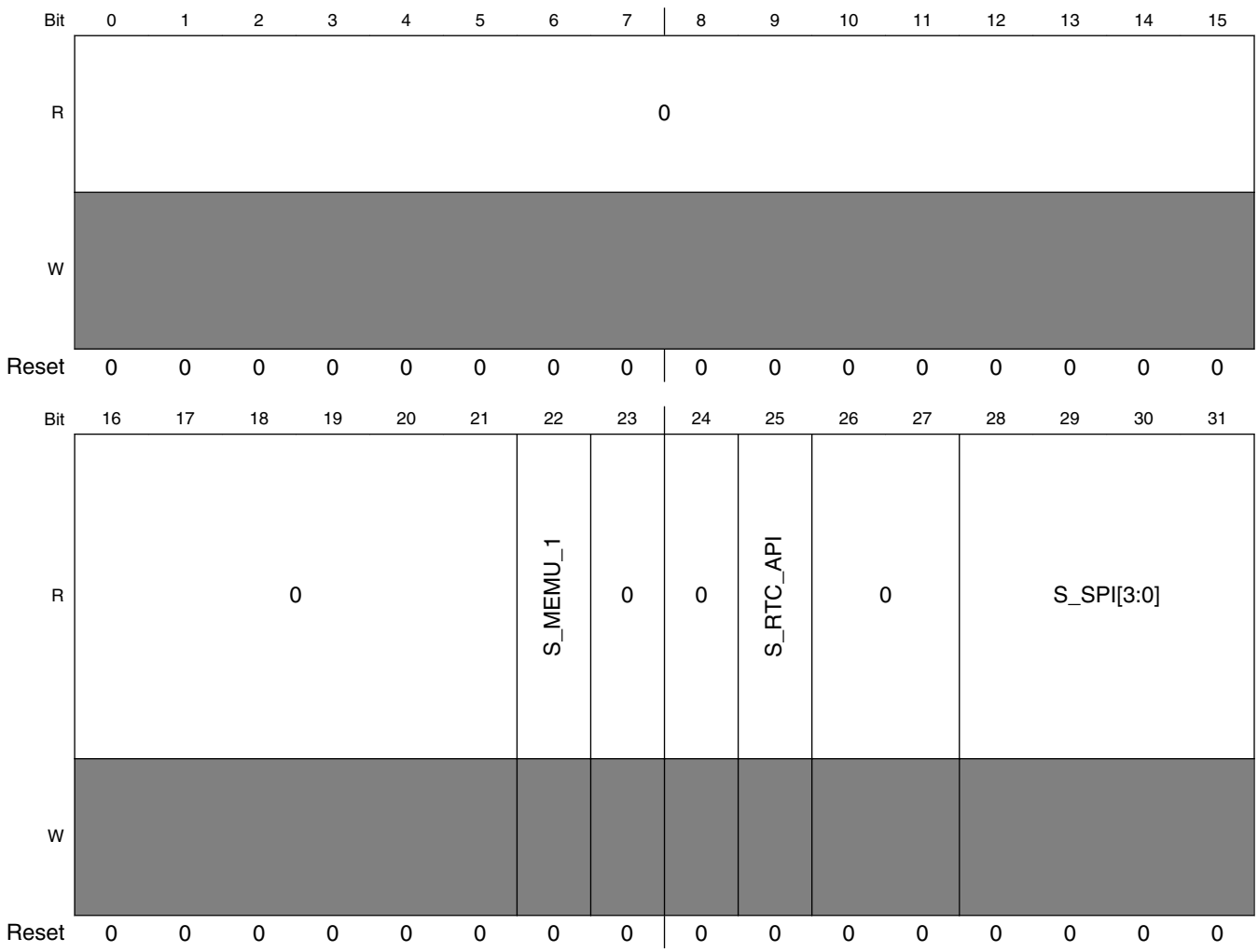
MC_ME_PS2 field descriptions (continued)

Field	Description
	1: Peripheral is active
3 S_LPU_CTL	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
4 S_PIT_RTI_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
5 S_DMAMUX	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0 Peripheral is frozen 1 Peripheral is active
6–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 S_CRC	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME.
12 S_MEMU_0	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME.
13 S_JDC	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME.
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–25 S_FLEXCAN[7:0]	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 S_LIN[15:14]	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active

40.3.20 Peripheral Status Register 3 (MC_ME_PS3)

This register provides the status of the peripherals.

Address: 0h base + 6Ch offset = 6Ch



MC_ME_PS3 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 S_MEMU_1	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active

Table continues on the next page...

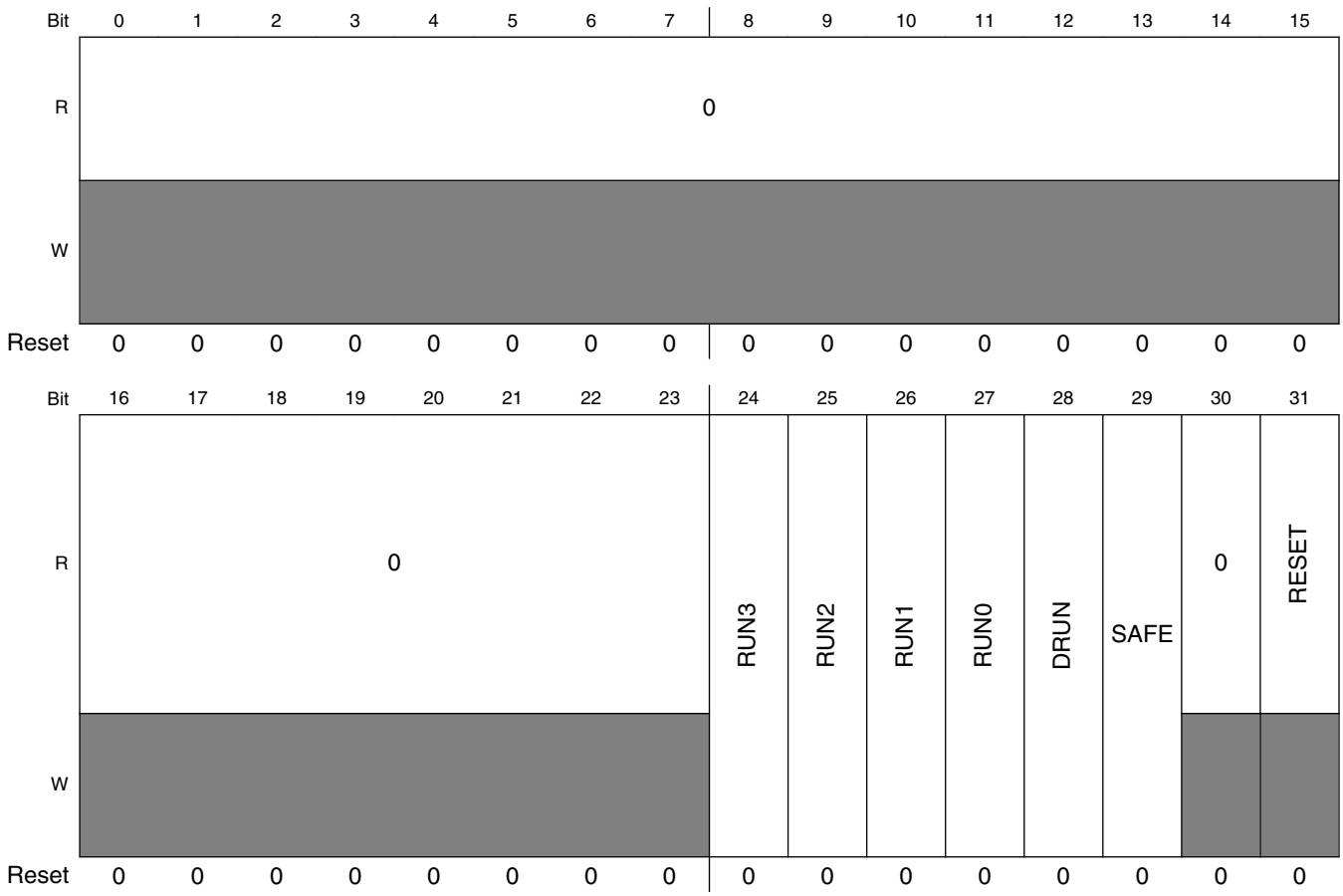
MC_ME_PS3 field descriptions (continued)

Field	Description
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 S_RTC_API	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 S_SPI[3:0]	Peripheral status — These bits specify the current status of each peripheral which is controlled by the MC_ME. 0: Peripheral is frozen 1: Peripheral is active

40.3.21 Run Peripheral Configuration Register (MC_ME_RUN_PC*n*)

The ME_RUN_PC0..7 registers configure eight different types of peripheral behavior during run modes.

Address: 0h base + 80h offset + (4d × i), where i=0d to 7d



MC_ME_RUN_PC*n* field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 RUN3	Peripheral control during RUN3 0 Peripheral is frozen with clock gated 1 Peripheral is active
25 RUN2	Peripheral control during RUN2 0 Peripheral is frozen with clock gated 1 Peripheral is active

Table continues on the next page...

MC_ME_RUN_PC*n* field descriptions (continued)

Field	Description
26 RUN1	Peripheral control during RUN1 0 Peripheral is frozen with clock gated 1 Peripheral is active
27 RUN0	Peripheral control during RUN0 0 Peripheral is frozen with clock gated 1 Peripheral is active
28 DRUN	Peripheral control during DRUN 0 Peripheral is frozen with clock gated 1 Peripheral is active
29 SAFE	Peripheral control during SAFE 0 Peripheral is frozen with clock gated 1 Peripheral is active
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RESET	Peripheral control during RESET 0 Peripheral is frozen with clock gated 1 Peripheral is active

40.3.22 Low-Power Peripheral Configuration Register (MC_ME_LP_PC*n*)

The MC_LP_PC registers configure eight different types of peripheral behavior during non-run modes.

Address: 0h base + A0h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			0			0		0							
W			STANDBY0			STOP0		Reserved								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MC_ME_LP_PC*n* field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 STANDBY0	Peripheral control during STANDBY0 0 Peripheral is frozen with clock gated 1 Peripheral is active
19–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 STOP0	Peripheral control during STOP0 0 Peripheral is frozen with clock gated 1 Peripheral is active
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 Reserved	This field is reserved.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

40.3.23 BCTU Peripheral Control Register (MC_ME_PCTL0)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL0 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.

Table continues on the next page...

MC_ME_PCTL0 field descriptions (continued)

Field	Description
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.24 eMIOS_0 Peripheral Control Register (MC_ME_PCTL1)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + C1h offset = C1h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL1 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses.

Table continues on the next page...

MC_ME_PCTL1 field descriptions (continued)

Field	Description
	0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.25 eMIOS_1 Peripheral Control Register (MC_ME_PCTL2)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + C2h offset = C2h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL2 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode

Table continues on the next page...

MC_ME_PCTL2 field descriptions (continued)

Field	Description
	NOTE: This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.26 SAI0 Peripheral Control Register (MC_ME_PCTL10)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + CAh offset = CAh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL10 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_ME_PCTL10 field descriptions (continued)

Field	Description
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.27 SAI1 Peripheral Control Register (MC_ME_PCTL11)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + CBh offset = CBh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL11 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.28 SAI2 Peripheral Control Register (MC_ME_PCTL12)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL12 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.29 ENET Peripheral Control Register (MC_ME_PCTL15)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + CFh offset = CFh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL15 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.30 CMP_0 Peripheral Control Register (MC_ME_PCTL20)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL20 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.31 CMP_1 Peripheral Control Register (MC_ME_PCTL21)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + D5h offset = D5h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL21 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.32 CMP_2 Peripheral Control Register (MC_ME_PCTL22)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + D6h offset = D6h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL22 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.33 ADC_0 Peripheral Control Register (MC_ME_PCTL24)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + D8h offset = D8h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL24 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.34 ADC_1 Peripheral Control Register (MC_ME_PCTL25)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + D9h offset = D9h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL25 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.35 FlexRay Peripheral Control Register (MC_ME_PCTL28)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL28 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.36 IIC_0 Peripheral Control Register (MC_ME_PCTL30)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + DEh offset = DEh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL30 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.37 IIC_1 Peripheral Control Register (MC_ME_PCTL31)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + DFh offset = DFh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL31 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.38 IIC_2 Peripheral Control Register (MC_ME_PCTL32)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + E0h offset = E0h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL32 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.39 IIC_3 Peripheral Control Register (MC_ME_PCTL33)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + E1h offset = E1h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL33 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.40 DSPI_0 Peripheral Control Register (MC_ME_PCTL40)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL40 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.41 DSPI_1 Peripheral Control Register (MC_ME_PCTL41)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + E9h offset = E9h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL41 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.42 DSPI_2 Peripheral Control Register (MC_ME_PCTL42)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + EAh offset = EAh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL42 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.43 DSPI_3 Peripheral Control Register (MC_ME_PCTL43)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + EBh offset = EBh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL43 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.44 LIN_0 Peripheral Control Register (MC_ME_PCTL50)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + F2h offset = F2h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL50 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.45 LIN_1 Peripheral Control Register (MC_ME_PCTL51)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + F3h offset = F3h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL51 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.46 LIN_2 Peripheral Control Register (MC_ME_PCTL52)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + F4h offset = F4h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL52 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.47 LIN_3 Peripheral Control Register (MC_ME_PCTL53)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + F5h offset = F5h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL53 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.48 LIN_4 Peripheral Control Register (MC_ME_PCTL54)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + F6h offset = F6h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL54 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.49 LIN_5 Peripheral Control Register (MC_ME_PCTL55)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + F7h offset = F7h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL55 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.50 LIN_6 Peripheral Control Register (MC_ME_PCTL56)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL56 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.51 LIN_7 Peripheral Control Register (MC_ME_PCTL57)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + F9h offset = F9h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL57 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.52 LIN 8 Peripheral Control Register (MC_ME_PCTL58)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + FAh offset = FAh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL58 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.53 LIN_9 Peripheral Control Register (MC_ME_PCTL59)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + FBh offset = FBh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL59 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.54 LIN_10 Peripheral Control Register (MC_ME_PCTL60)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + FCh offset = FCh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL60 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.55 LIN_11 Peripheral Control Register (MC_ME_PCTL61)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + FDh offset = FDh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL61 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.56 LIN 12 Peripheral Control Register (MC_ME_PCTL62)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + FEh offset = FEh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL62 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.57 LIN_13 Peripheral Control Register (MC_ME_PCTL63)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + FFh offset = FFh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL63 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.58 LIN 14 Peripheral Control Register (MC_ME_PCTL64)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 100h offset = 100h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL64 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.59 LIN_15 Peripheral Control Register (MC_ME_PCTL65)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 101h offset = 101h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL65 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.60 FLEXCAN_0 Peripheral Control Register (MC_ME_PCTL70)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 106h offset = 106h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL70 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.61 FLEXCAN_1 Peripheral Control Register (MC_ME_PCTL71)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 107h offset = 107h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL71 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.62 FLEXCAN_2 Peripheral Control Register (MC_ME_PCTL72)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 108h offset = 108h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL72 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.63 FLEXCAN_3 Peripheral Control Register (MC_ME_PCTL73)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 109h offset = 109h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL73 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.64 FLEXCAN_4 Peripheral Control Register (MC_ME_PCTL74)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 10Ah offset = 10Ah

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL74 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.65 FLEXCAN_5 Peripheral Control Register (MC_ME_PCTL75)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 10Bh offset = 10Bh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL75 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.66 FLEXCAN_6 Peripheral Control Register (MC_ME_PCTL76)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 10Ch offset = 10Ch

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL76 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.67 FLEXCAN_7 Peripheral Control Register (MC_ME_PCTL77)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 10Dh offset = 10Dh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL77 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.68 JDC Control Register (MC_ME_PCTL82)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 112h offset = 112h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL82 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.69 MEMU_0 Control Register (MC_ME_PCTL83)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 113h offset = 113h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL83 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.70 CRC Control Register (MC_ME_PCTL84)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 114h offset = 114h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL84 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.71 DMAMUX Control Register (MC_ME_PCTL90)

This register selects the configurations during run mode for the associated peripheral. There is no affect in low power modes.

Address: 0h base + 11Ah offset = 11Ah

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL90 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.72 PIT_RTI_0 Control Register (MC_ME_PCTL91)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 11Bh offset = 11Bh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL91 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.73 WKPU Control Register (MC_ME_PCTL93)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 11Dh offset = 11Dh

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL93 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.74 SPI0 Control Register (MC_ME_PCTL96)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 120h offset = 120h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL96 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.75 SPI1 Control Register (MC_ME_PCTL97)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 121h offset = 121h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL97 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.76 SPI2 Control Register (MC_ME_PCTL98)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 122h offset = 122h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL98 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.77 SPI3 Control Register (MC_ME_PCTL99)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 123h offset = 123h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL99 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.78 RTC-API Control Register (MC_ME_PCTL102)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 126h offset = 126h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

MC_ME_PCTL102 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.79 MEMU_1 Control Register (MC_ME_PCTL105)

This register selects the configurations during run and non-run modes for the associated peripheral.

Address: 0h base + 129h offset = 129h

Bit	0	1	2	3	4	5	6	7
Read	0	DBG_F	LP_CFG			RUN_CFG		
Write								
Reset	0	0	0	0	0	0	0	0

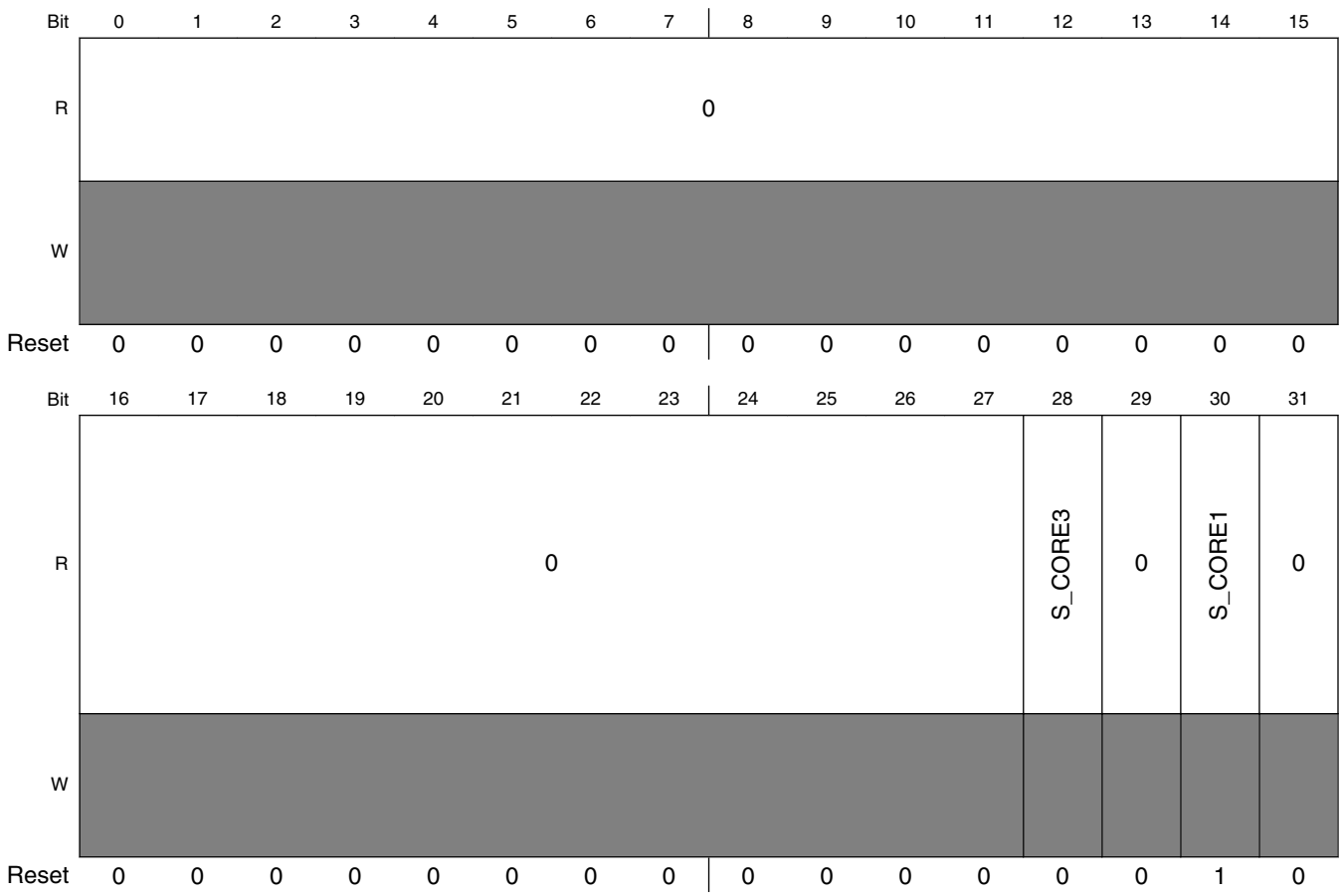
MC_ME_PCTL105 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 DBG_F	Peripheral control in debug mode — This bit controls the state of the peripheral in debug mode **NOTE:** This feature is useful to freeze the peripheral state while entering debug. For example, this may be used to prevent a reference timer from running while making a debug accesses. 0 Peripheral state depends on RUN_CFG/LP_CFG bits and the chip mode 1 Peripheral is frozen if not already frozen in chip modes.
2–4 LP_CFG	Peripheral configuration select for non-run modes — These bits associate a configuration as defined in the ME_LP_PC0…7 registers to the peripheral. 000 Selects ME_LP_PC0 configuration 001 Selects ME_LP_PC1 configuration 010 Selects ME_LP_PC2 configuration 011 Selects ME_LP_PC3 configuration 100 Selects ME_LP_PC4 configuration 101 Selects ME_LP_PC5 configuration 110 Selects ME_LP_PC6 configuration 111 Selects ME_LP_PC7 configuration
5–7 RUN_CFG	Peripheral configuration select for run modes — These bits associate a configuration as defined in the ME_RUN_PC0…7 registers to the peripheral. 000 Selects ME_RUN_PC0 configuration 001 Selects ME_RUN_PC1 configuration 010 Selects ME_RUN_PC2 configuration 011 Selects ME_RUN_PC3 configuration 100 Selects ME_RUN_PC4 configuration 101 Selects ME_RUN_PC5 configuration 110 Selects ME_RUN_PC6 configuration 111 Selects ME_RUN_PC7 configuration

40.3.80 Core Status Register (MC_ME_CS)

This register provides the status of each core.

Address: 0h base + 1C0h offset = 1C0h



MC_ME_CS field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 S_CORE3	Core3 Status These bits specify the current status of Z2 core which is controlled by the MC_ME. 0 Z2 is disabled 1 Z2 is running
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 S_CORE1	Core 1 status — These bits specify the current status of Z4A core which is controlled by the MC_ME.

Table continues on the next page...

MC_ME_CS field descriptions (continued)

Field	Description
	0 Z4A is disabled 1 Z4A is running
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

40.3.81 Core1 Control Register (MC_ME_CCTL1)

This register controls whether the Z4A is disabled or running during run modes. Please refer to memory map table for information on which ME_CCTL1 locations are actually occupied. The unoccupied locations contain a read-only half-word value of 0x0000.

The run mode bits (SAFE, DRUN, and RUNx) of the ME_CCTL1 register are pre-loaded with the value provided by the device configuration just before reset exist.

The value of the DRUN bit just before entering STANDBY0 mode is the value of all the run mode bits (SAFE, DRUN, and RUNx) on STANDBY0 exit. The values of the low-power mode bits (STOP0, and STANDBY0) revert to their reset values on STANDBY0 exit.

The low-power mode bits (STOP0, and STANDBY0) of the ME_CCTL1 register are reset with the default value shown and are not pre-loaded with another value.

Address: 0h base + 1C6h offset = 1C6h

Bit	0	1	2	3	4	5	6	7
Read	0	STANDBY0		0		STOP0	0	0
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	0	RESET
Write								
Reset	0	0	0	1	1	0	0	0

MC_ME_CCTL1 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–2 STANDBY0	Core control during STANDBY0 0 Z4A is disabled with clock gated

Table continues on the next page...

MC_ME_CCTL1 field descriptions (continued)

Field	Description
3–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 — Z4A is disabled with clock gated.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 RUN3	Core control during RUN3 0 Z4A is disabled with clock gated 1 Z4A is running
9 RUN2	Core control during RUN2 0 Z4A is disabled with clock gated 1 Z4A is running
10 RUN1	Core control during RUN1 0 Z4A is disabled with clock gated 1 Z4A is running
11 RUN0	Core control during RUN0 0 Z4A is frozen with clock gated 1 Z4A is running
12 DRUN	Core control during DRUN 0 Z4A is frozen with clock gated 1 Z4A is running
13 SAFE	Core control during SAFE 0 Z4A is frozen with clock gated 1 Z4A is running
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 RESET	Core control during RESET — The Z4A is always disabled during RESET.

40.3.82 Core3 Control Register (MC_ME_CCTL3)

This register controls whether the Z2 core is disabled or running during run modes.

The run mode bits (SAFE, DRUN, and RUNx) of the ME_CCTL3 register are pre-loaded with the value provided by the device configuration just before reset exist.

The value of the DRUN bit just before entering STANDBY0 mode is the value of all the run mode bits (SAFE, DRUN, and RUNx) on STANDBY0 exit. The values of the low-power mode bits (STOP0, and STANDBY0) revert to their reset values on STANDBY0 exit. The low-power mode bits (STOP0, and STANDBY0) of the ME_CCTL3 register are reset with the default value shown and are not pre-loaded with another value.

Address: 0h base + 1CAh offset = 1CAh

Bit	0	1	2	3	4	5	6	7
Read	0	STANDBY0			0	STOP0	0	0
Write								
Reset	0*	0*	0*	0*	0*	0*	0*	0*

Bit	8	9	10	11	12	13	14	15
Read	RUN3	RUN2	RUN1	RUN0	DRUN	SAFE	0	RESET
Write								
Reset	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value of RUN3, RUN2, RUN1, RUN0, DRUN, and SAFE are determined by the chip configuration and boot mode.

MC_ME_CCTL3 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–2 STANDBY0	Core control during STANDBY0 0 Z2 is disabled with clock gated
3–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 STOP0	Core control during STOP0 — Z2 is disabled with clock gated.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 RUN3	Core control during RUN3 0 Z2 is disabled with clock gated 1 Z2 is running
9 RUN2	Core control during RUN2 0 Z2 is disabled with clock gated 1 Z2 is running
10 RUN1	Core control during RUN1 0 Z2 is disabled with clock gated 1 Z2 is running

Table continues on the next page...

MC_ME_CCTL3 field descriptions (continued)

Field	Description
11 RUN0	Core control during RUN0 0 Z2 is frozen with clock gated 1 Z2 is running
12 DRUN	Core control during DRUN 0 Z2 is frozen with clock gated 1 Z2 is running
13 SAFE	Core control during SAFE 0 Z2 is frozen with clock gated 1 Z2 is running
14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 RESET	Core control during RESET — The Z2 is always disabled during RESET.

40.3.83 CORE1 Address Register (MC_ME_CADDR1)

This register contain the boot address of each Z4A and a bit for controlling whether the Z4A core is to be reset on the next mode change that has the core configured to be running in the target mode.

This register can be written only as words and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = ‘1’). A write access to any of this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

ME_CADDR1 is pre-loaded with the value provided by the device configuration just before reset exit in order to ensure that Z4A boots from the correct address after any chip reset if the DRUN bit in the ME_CCTL1 register has a default value of ‘1’.

The value of the ADDR field just before entering STANDBY0 mode is the value of the ADDR field on STANDBY0 exit.

Address: 0h base + 1E4h offset = 1E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDR														0	RMC
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0

* Notes:

- Reset value of ADDR field is determined by the chip configuration and boot mode.

MC_ME_CADDR1 field descriptions

Field	Description
0–29 ADDR	Core Address — This field is used by Z4A cores as the boot address (32-bit word aligned) when Z4A next exits reset.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RMC	Reset on Mode Change The Z4A will be reset on the next mode change that has the core configured to be running in the target mode. 0 Z4A will not be reset on the next mode change 1 Z4A will be reset on the next mode change that has the core configured to be running in the target mode. This field is automatically cleared when the core reset triggered by the RMC is executed.

40.3.84 CORE3 Address Register (MC_ME_CADDR3)

This register contain the boot address of Z2 core and a bit for controlling whether the corresponding core is to be reset on the next mode change that has the Z2 core configured to be running in the target mode.

This register can be written only as words and cannot be written after a mode change request has been made until the mode transition has completed (i.e., while the S_MTRANS bit of the ME_GS register = ‘1’). A write access to this register during this time will result in the ICONF_CC flag in the ME_IS register being asserted.

ME_CADDR3 is pre-loaded with the value provided by the device configuration just before reset exit in order to ensure that Z2 core boots from the correct address after any chip reset if the DRUN bit in the ME_CCTL3 register has a default value of ‘1’.

The value of the ADDR field just before entering STANDBY0 mode is the value of the ADDR field on STANDBY0 exit.

Address: 0h base + 1ECh offset = 1ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADDR															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ADDR														0	RMC
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- Reset value is determined by the chip configuration and boot mode.

MC_ME_CADDR3 field descriptions

Field	Description
0–29 ADDR	Core Address — This field is used by Z2 as the boot address (32-bit word aligned) when next exits reset.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RMC	Reset on Mode Change The Z2 will be reset on the next mode change that has the core configured to be running in the target mode. 0 Z2 will not be reset on the next mode change 1 Z2 will be reset on the next mode change that has the core configured to be running in the target mode. This field is automatically cleared when the core reset triggered by the RMC field is executed.

40.4 Functional Description

40.4.1 Mode Transition Request

The transition from one mode to another mode is normally handled by software by accessing the mode control register ME_MCTL. In case of special events, the mode transition can be automatically managed by hardware. In order to switch from one mode to another, the application should access the ME_MCTL register twice by writing

- the first time with the value of the key (0x5AF0) into the KEY bit field and the required target mode into the TARGET_MODE bit field,
- and the second time with the inverted value of the key (0xA50F) into the KEY bit field and the required target mode into the TARGET_MODE bit field.

Once a valid mode transition request is detected, the target mode configuration information is loaded from the corresponding ME_<mode>_MC register.The mode transition request may require a number of cycles depending on the programmed configuration, and software should check the S_CURRENT_MODE bit field and the S_MTRANS bit of the global status register ME_GS to verify when the mode has been correctly entered and the transition process has completed. For a description of valid mode requests, please refer to Mode Transition Interrupts.

Any modification of the mode configuration register of the currently selected mode will not be taken into account immediately but on the next request to enter this mode. This means that transition requests such as RUN0…3 → RUN0…3, DRUN → DRUN, SAFE

→ SAFE are considered valid mode transition requests. As soon as the mode request is accepted as valid, the S_MTRANS bit is set till the status in the ME_GS register matches the configuration programmed in the respective ME_<mode>_MC register.

Note

Software should poll the ME_GS[S_MTRANS] field after requesting a transition to STOP0 or STANDBY0 modes or after reset exit and should initiate any transaction once the mode transition to the target mode is completed.

All mode transitions shown in figure, Mode Transition Interrupts are initiated by software except for those from RESET and the low-power modes or to SAFE and RESET on a hardware failure. See Table "MC_ME Mode Descriptions" for a more detailed summary of mode entry and exit triggers.

40.4.2 Modes Details

40.4.2.1 RESET Mode

The chip enters this mode on the following events:

- from SAFE or DRUN, RUN0…3 when the TARGET_MODE bit field of the ME_MCTL register is written with either "0000" for a 'functional' reset or "1111" for a 'destructive' reset
- from any mode due to a system reset by the MC_RGM because of some non-recoverable hardware failure in the system (see the MC_RGM chapter for details)

Transition to this mode is instantaneous, and the system remains in this mode until the reset sequence is finished. The mode configuration information for this mode is provided by the ME_RESET_MC register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock.

40.4.2.2 DRUN Mode

The chip enters this mode on the following events:

- automatically from RESET mode after completion of the reset sequence
- from RUN0…3 or SAFE when the TARGET_MODE bit field of the ME_MCTL register is written with "0011"

As soon as any of the above events has occurred, a DRUN mode transition request is generated. The mode configuration information for this mode is provided by the ME_DRUN_MC register. In this mode, the flash, all clock sources, and the system clock configuration can be controlled by software as required. After system reset, the software execution starts with the default configuration selecting the 16 MHz int. RC osc. as the system clock. The clock source configuration except system clock source 0 can be chosen by preloaded information in flash memory. This information is loaded into the ME_DRUN_MC register during PHASE3 of the reset sequence.

This mode is intended to be used by software

- to initialize all registers as per the system needs
- to execute small routines in a ‘ping-pong’ with the STANDBY0 mode

When this mode is entered from STANDBY0 after a wakeup event, the ME_DRUN_MC register content is restored to its pre-STANDBY0 values, and the mode starts in that configuration.

Note

On exit from STANDBY0 after a wakeup event, the software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

40.4.2.3 SAFE Mode

The chip enters this mode on the following events:

- from any mode except RESET when the TARGET_MODE bit field of the ME_MCTL register is written with "0010"
- from any mode except RESET due to a SAFE mode request generated by the MC_RGM because of some potentially recoverable hardware failure in the system (see the MC_RGM chapter for details)

Note

If a hardware SAFE mode request occurs during RESET, depending on the timing of the SAFE mode request, SAFE mode may be entered immediately after the normal completion

of the reset sequence or several system clock cycles after DRUN entry. The SAFE mode request does not have any influence on the execution of the reset sequence itself.

Note

Do not confuse Safe Mode with the Functional Safety description of 'A safe state'. A Safe State is determined by the application in accordance with ISO26262 and the mode entry module safe mode is an unrelated state which can be entered from software via the following descriptions.

As soon as any of the above events has occurred, a SAFE mode transition request is generated. The mode configuration information for this mode is provided by the ME_SAFE_MC register. This mode has a pre-defined configuration, and the 16 MHz int. RC osc. is selected as the system clock.

If the SAFE mode is requested by software while some other mode transition process is ongoing, the new target mode becomes the SAFE mode regardless of other pending requests or new requests during the mode transition. No new mode request made during a transition to the SAFE mode will cause an invalid mode interrupt.

Note

If software requests to change to the SAFE mode and then requests to change back to the parent mode before the mode transition is completed, the chip's final mode after mode transition will be the SAFE mode, and an invalid mode transition interrupt will be generated.

As long as a SAFE event is active, the system remains in the SAFE mode, and any software mode request during this time is ignored and lost.

This mode is intended to be used by software

- to assess the severity of the cause of failure and then to either
  - re-initialize the chip via the DRUN mode, or
  - completely reset the chip via the RESET mode.

If the outputs of the system I/Os need to be forced to a high impedance state upon entering this mode, the PDO bit of the ME_SAFE_MC register should be set. The input levels remain unchanged.

40.4.2.4 RUN0…3 Modes

The chip enters one of these modes on the following events:

- from the DRUN or another RUN0…3 mode when the TARGET_MODE bit field of the ME_MCTL register is written with "0100…0111"
- from the STOP0 mode due to an interrupt or wakeup event

As soon as any of the above events has occurred, a RUN0…3 mode transition request is generated. The mode configuration information for these modes is provided by the ME_RUN0…3_MC registers. In these modes, the flash, all clock sources, and the system clock configuration can be controlled by software as required.

These modes are intended to be used by software

- to execute application routines

Note

Software must ensure that the code executes from RAM before changing to this mode if the flash is configured to be in the low-power or power-down state in this mode.

40.4.2.5 STOP0 Mode

The chip enters this mode on the following events:

- from one of the RUN0…3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with "1010".

As soon as any of the above events has occurred, a STOP0 mode transition request is generated. The mode configuration information for this mode is provided by the ME_STOP0_MC register. This mode is fully configurable, and the ME_STOP0_MC register should be programmed according to the system needs. The following clock sources are switched off in this mode:

- the PLL PHI0

The flash memories can be put in power-down mode as needed. If there is a STOP0 mode request when any interrupt or wakeup event is active, the transition to STOP0 is aborted with the resultant mode being the current mode, and an invalid mode transition interrupt

is generated and ME_IMTS.MRIG bit is set. If a transition to safe mode is in progress, a transition to STOP0 mode will also set the ME_IMTS.MRIG bit. If a transiton to STOP0 mode is interrupted by a transition to DRUN (on reset), an invalid mode interrupt is not generated.

This can be used as an advanced low-power mode with the core clock frozen and almost all peripherals stopped.

This mode is intended as an advanced low-power mode with

- the core clock frozen
- the additional core clocks frozen
- almost all peripherals stopped

and to be used by software

- to wait until it is required to do something with no need to react quickly (e.g., allow for system clock source to be re-started)

The normal exit from the STOP0 mode is by an interrupt or a wakeup event.

This mode can be used to stop all clock sources and thus preserve the chip status. When exiting the STOP0 mode, the 16 MHz internal RC oscillator clock is selected as the system clock until the target clock is available.

Note

It is good practice for software to ensure that the S_MTRANS bit in the ME_GS register has been cleared on STOP0 mode exit to ensure that the previous RUN0…3 mode configuration has been fully restored before executing critical code.

NOTE

If waking up from stop mode through module interrupt and module is configured to be OFF in RUN and ON in STOP mode, then on stop exit, ME_GS may not show MTRANS as low till the module interrupt is cleared in ISR and so to avoid code hanging on MTRANS poll, if added as first check on stop exit, the MTRANS polling has to be done after clearing module interrupt.

NOTE

All interrupts that are masked in INTC should be disabled in the modules as well. Otherwise a hang situation can occur as follows:

- Module issues an interrupt and STOP_ACK to MC_ME is gated.
- MC_ME does not retract and wait for STOP_ACK.
- This interrupt is not available at INTC so the core cannot service it, which causes a hang.

NOTE

A potential system hang scenario may occur if a stop mode request is generated when the interrupts are enabled in active modules and interrupts appears when the MC_ME is transitioning to stop mode. User needs to adhere to some software guidelines:

- Disable Interrupts before requesting a stop mode change.
- If the previous guideline is not possible ensure any interrupt are finished before stop mode change.

40.4.2.6 STANDBY0 Mode

The chip enters this mode on the following events:.

- from the DRUN or one of the RUN0…3 modes when the TARGET_MODE bit field of the ME_MCTL register is written with “1101”.

As soon as any of the above events occur, a STANDBY0 mode transition request is generated. The mode configuration information for this mode is provided by the ME_STANDBY0_MC register. In this mode, the power supply is turned off for most of the chip. The only parts of the chip that are still powered during this mode are pads mapped on wakeup lines and power domain #0 which contains the MC_RGM, MC_PCU, WKPU etc.. The IRC can be optionally switched off. This is the lowest power consumption mode possible on the chip.

NOTE

Please ensure before entering STANDBY0 Mode, PMCDIG_MCR[LVD_PD2_COLD_REE] bit is programmed to 0.This will ensure LVD_PD2_COLD as source of functional reset doesnt get triggered on STANDBY0 Mode entry.Please refer PMCDIG chapter for further details.

This mode is intended as an extreme low-power mode with

- the cores, the flashes, and almost all peripherals and memories powered down

and to be used by software

- to wait until it is required to do something with no need to react quickly (i.e., allow for system power-up and system clock source to be re-started)

The normal exit of the STANDBY0 mode is from a wakeup event. Interrupt events will not cause STANDBY0 mode exit.

The exit sequence of this mode is similar to the reset sequence, and the resets to all but power domain #0 are asserted with timing that is the same as reset PHASE1 through PHASE3. However, in addition to booting from the default location, the chip can also be configured to boot from the backup RAM. In the case of booting from backup RAM , it is also possible to keep the flashes disabled by writing “01” to the FLAON fields in the ME_DRUN_MC register prior to STANDBY0 entry.

If there is a STANDBY0 mode request while any wakeup event is active, the chip mode does not change.

All power domains except power domains #0 and #1 are configurable in this mode in order to reduce leakage consumption. Active power domains are determined by the power configuration registers of the MC_PCU.

40.4.3 Mode Transition Process

The process of mode transition follows the following steps in a pre-defined manner depending on the current chip mode and the requested target mode. In many cases of mode transition, not all steps need to be executed based on the mode control information, and some steps may not be applicable according to the mode definition itself.

Since the MC_ME ensures that all resources are in the proper state during the mode transition process, the following, for example, do not need to be checked by software.

- When the 8-40 MHz crystal oscillator is enabled in a mode configuration register, a mode transition to that mode will not complete until the End Of Count timeout occurs for the oscillator. Software does not need to interrogate the End Of Count timeout status bit.
- When a PLL is enabled in a mode configuration register, a mode transition to that mode will not complete until the PLL is locked. Software does not need to interrogate the PLL’s lock status bit.

40.4.3.1 Target Mode Request

The target mode is requested by accessing the ME_MCTL register with the required keys. This mode transition request by software must be a valid request satisfying a set of pre-defined rules to initiate the process. If the request fails to satisfy these rules, it is ignored, and the TARGET_MODE bit field is not updated. An optional interrupt can be generated for invalid mode requests. Refer to Mode Transition Interrupts for details.

In the case of mode transitions occurring because of hardware events such as a reset, a SAFE mode request, or interrupt requests and wakeup events to exit from low-power modes, the TARGET_MODE bit field of the ME_MCTL register is automatically updated with the appropriate target mode. The mode change process start is indicated by the setting of the mode transition status bit S_MTRANS of the ME_GS register.

A RESET mode requested via the ME_MCTL register is passed to the MC_RGM, which generates a global system reset and initiates the reset sequence. The RESET mode request has the highest priority, and the MC_ME is kept in the RESET mode during the entire reset sequence.

The SAFE mode request has the next highest priority after reset. It can be generated either by software via the ME_MCTL register from all software running modes or by the MC_RGM after the detection of system hardware failures, which may occur in any mode.

40.4.3.2 Target Mode Configuration Loading

On completion of the Target Mode Request step, the target mode configuration from the ME_<target mode>_MC register is loaded to start the resources (voltage sources, clock sources, flash, pads, etc.) control process.

An overview of resource control possibilities for each mode is shown in the following table. A '√' indicates that a given resource is configurable for a given mode.

Table 40-8. MC_ME Resource Control Overview

Resource	Mode					
	RESET	SAFE	DRUN	RUN0...3	STOP0	STANDBY0
IRC	on	on	on	on	√ on	√ on
FXOSC	off	off	√ off	√ off	√ off	√ off
PLL	off	off	√ off	√ off	off	off
SIRC			√	√	√	√

Table continues on the next page...

Table 40-8. MC_ME Resource Control Overview (continued)

Resource	Mode					
	RESET	SAFE	DRUN	RUN0…3	STOP0	STANDBY0
	off	off	on	off	off	off
SXOSC	off	off	√ off	√ off	√ off	√ off
FLASH	normal	normal	√ normal	√ normal	√ power-down	power-down
MVREG	on	on	on	on	on	off
PDO	off	√ on	off	off	√ off	on

40.4.3.3 Peripheral Clocks Disable

On completion of the Target Mode Request step and the system clock frequency ramp-down of the System Clock Switching step, the MC_ME requests each peripheral to enter its stop mode when:

- the peripheral is configured to be disabled via the target mode, the peripheral configuration registers ME_RUN_PC0…7 and ME_LP_PC0…7, and the peripheral control registers ME_PCTL***n***

Note

The MC_ME automatically requests peripherals to enter their stop modes if the power domains in which they are residing are to be turned off due to a mode change. However, it is good practice for software to ensure that those peripherals that are to be powered down are configured in the MC_ME to be frozen.

Each peripheral acknowledges its stop mode request after closing its internal activity. The MC_ME then disables the corresponding clock(s) to this peripheral.

In the case of a SAFE mode transition request, the MC_ME does not wait for the peripherals to acknowledge the stop requests. The SAFE mode clock gating configuration is applied immediately regardless of the status of the peripherals' stop acknowledges.

Please refer to Peripheral Clock Gating for more details.

Each peripheral that may block or disrupt a communication bus to which it is connected ensures that these outputs are forced to a safe or recessive state when the chip enters the SAFE mode.

40.4.3.4 Processor Low-Power Mode Entry

If, on completion of the Peripheral Clocks Disable step, the mode transition is to the STOP0 or STANDBY0 mode, the MC_ME requests all processors which are configured in ME_CCTL***n*** to be disabled to enter its stopped state. Each processor acknowledges its stop state request bit after completing all outstanding bus transactions.

If, on completion of the Peripheral Clocks Disable step, the mode transition is from a non-low-power mode to another non-low-power mode, the MC_ME requests each processor which is configured in the ME_CCTL***n*** registers to be disabled to enter its stopped state. The processor acknowledges its stop state request bit after completing all outstanding bus transactions.

40.4.3.5 Processor Clocks Disable

If, on completion of the Processor Low-Power Mode Entry step, and all processors which are configured in the ME_CCTL***n*** registers to be disabled are in their appropriate stopped state, the MC_ME disables the applicable processor clocks bits to achieve further power saving.

The clocks to processors which are configured to be running in both the current and target modes or disabled in both the current and target modes are unaffected while transitioning between non-low-power modes.

Warning

Clocks to the whole chip including the processor and system memory can be disabled in TEST mode.

40.4.3.6 System Clock Disable

If, on completion of the Processor Clocks Disable step, all processor clocks have been disabled, the system clock used for non-processor system functions and memory is disabled.

40.4.3.7 Clock Sources (Main Voltage Regulator Independent) Switch-On

On completion of the Processor Low-Power Mode Entry step, the MC_ME switches on all clock sources based on the <clock source>ON bits of the ME_<current mode>_MC and ME_<target mode>_MC registers. The following clock sources are switched on at this step:

- the 16 MHz internal RC oscillator
- the 8-40 MHz crystal oscillator

NOTE

Clock sources which need the main voltage regulator to be stable are not controlled by this step.

The clock sources that are required by the target mode are switched on. The duration required for the output clocks to be stable depends on the type of source, and all further steps of mode transition depending on one or more of these clocks waits for the stable status of the respective clocks. The availability status of these clocks is updated in the S_<clock source> bits of ME_GS register.

The clock sources which need to be switched off are unaffected during this process in order to not disturb the system clock which might require one of these clocks before switching to a different target clock.

40.4.3.8 Main Voltage Regulator Switch-On

On completion of the Target Mode Request step, step, if the main voltage regulator needs to be switched on from its off state based on the MVRON bit of the ME_CURRENT MODE_MC and ME_TARGET MODE_MC registers, the MC_ME requests the MC_PCU to power-up the regulator and waits for the output voltage stable status in order to update the S_MVR bit of the ME_GS register.

This step is required only during the exit of the low-power modes STOP0. In this step, the 16 MHz internal RC oscillator is switched on regardless of the target mode configuration, as the main voltage regulator requires the 16 MHz int. RC osc. during power-up in order to generate the voltage status.

During the STANDBY0 exit sequence, the MC_PCU alone manages the power-up of the main voltage regulator, and the MC_ME is kept in RESET or shut off (depending on the power domain #1 status).

40.4.3.9 Flash Module Switch-On

On completion of the step, if the flash needs to be switched to normal mode from its low-power or power-down mode based on the FLAON bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, the MC_ME requests the flash to exit from its low-power/power-down mode. When the flash is available for access, the S_FLA bit field of the ME_GS register is updated to "11" by hardware.

Warning

It is illegal to switch the FLASH from low-power mode to power-down mode and from power-down mode to low-power mode. The MC_ME, however, does not prevent this nor does it flag it.

40.4.3.10 Clock Sources (Main Voltage Regulator Dependent) Switch-On

On completion of the Clock Sources (Main Voltage Regulator Independent) Switch-On and Main Voltage Regulator Switch-On,the MC_ME controls all clock sources, which need the main voltage regulator to be on, based on the CLOCK SOURCES ON bits of the ME_CURRENT_MODE_MC and ME_TARGET_MODE_MC registers. The following clock sources are switched on at this step:

- the PLL PHI_0

40.4.3.11 Pad Outputs-On

On completion of the Main Voltage Regulator Switch-On step, if the PDO bit of the ME_<target mode>_MC register is cleared, then

- all pad outputs are enabled to return to their previous state
- the I/O pads power sequence driver is switched on

40.4.3.12 Peripheral Clocks Enable

Based on the current and target chip modes, the peripheral configuration registers ME_RUN_PC0…7, ME_LP_PC0…7, and the peripheral control registers ME_PCTLn, the MC_ME enables the clocks for selected modules as required. This step is executed only after the Main Voltage Regulator Switch-On process is completed, and if the value of the PWRLVL field of the ME_TARGET_MODE_MC register is different from that of the ME_current mode_MC register's, the peripheral clock enable/disable isn't updated until after the system clock frequency ramp-down step of the System Clock Switching process is completed.

Also, if a mode change translates to a power up of one or more power domains, the MC_PCU indicates the MC_ME after completing the power-up sequence upon which the MC_ME may assert the peripheral clock enables of the peripherals residing in those power domains.

40.4.3.13 System and Processor Clocks Enable

On completion of the Flash Module Switch-On step, the MC_ME enables the non-processor system clock and the clocks of all processors that are configured in the ME_CCTL***n*** registers to be disabled in the current mode and to be running in the target mode.

If the value of the PWRLVL field of the ME_<target mode>_MC register is different from that of the ME_<current mode>_MC register, the mode change handshake will not be initiated until after the system clock frequency ramp-down step of the System Clock Switching process is completed.

40.4.3.14 Processor Low-Power Mode Exit

On completion of the System and Processor Clocks Enable step, the MC_ME requests all processors which are configured in the ME_CCTL***n*** registers to be disabled in the current mode and to be running in the target mode.

40.4.3.15 System Clock Switching

Based on the SYSCLK bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the target and current system clock configurations differ, the following method is implemented for clock switching.

- The target clock configuration for the 16 MHz int. RC osc. takes effect only after the S_IRC bit of the ME_GS register is set by hardware (i.e., the 16 MHz internal RC oscillator has stabilized).
- The target clock configuration for the FXOSC takes effect only after the S_FXOSC bit of the ME_GS register is set by hardware (8-40 MHz crystal oscillator has been stabilized).
- If the clock is to be disabled, the SYSCLK bit field should be programmed with “1111”. In the STANDBY0 mode, the clock configuration is fixed, and the system clock is automatically forced to ‘0’.

If the value of the PWRLVL field of the ME_<target mode>_MC register not equal to that of the ME_<current mode>_MC register's, a progresive clock switching is performed in the power consumption of the device.

The current system clock configuration can be observed by reading the S_SYSCLK bit field of the ME_GS register, which is updated after every system clock switching. Until the target clock is available, the system uses the previous clock configuration.

System clock switching starts only after

- the Clock Sources (Main Voltage Regulator Independent) Switch-On process has completed if the target system clock source is one of the following:
  - the PLL PHI_0
- the Peripheral Clocks Disable process has completed in order not to change the system clock frequency before peripherals close their internal activitiesthe Peripheral Clocks Enable processe has completed

An overview of system clock source selection possibilities for each mode is shown in the following table. A '√' indicates that a given clock source is selectable for a given mode. A shaded cell indicates that the given clock source is not selectable for a given mode.

Table 40-9. MC_ME System Clock Selection Overview

System Clock Source	Mode					
	RESET	SAFE	DRUN	RUN0…3	STOP0	STANDBY0
16 MHz int. RC osc.	√ (default)	√ (default)	√ (default)	√ (default)	√ (default)	
FXOSC			√	√	√	
PLL PHI_0			√	√	√	
SIRC			√	√		
SXOSC			√	√		

Table continues on the next page...

Table 40-9. MC_ME System Clock Selection Overview (continued)

System Clock Source	Mode					
	RESET	SAFE	DRUN	RUN0...3	STOP0	STANDBY0
system clock is disabled					√	√ (default)

40.4.3.16 Pad Switch-Off

If the PDO bit of the ME_<target mode>_MC register is '1' then

- the outputs of the pads are forced to the high impedance state if the target mode is SAFE or TEST.

In STANDBY0 mode, the power sequence driver and all pads except the external reset and those mapped on wakeup lines are not powered and therefore high impedance. The wakeup line configuration remains unchanged.

This step is executed only after the Peripheral Clocks Disable process has completed.

40.4.3.17 Clock Sources Switch-Off

Based on the chip mode and the <clock source>ON bits of the ME_<mode>_MC registers, if a given clock source is to be switched off, the MC_ME requests the clock source to power down and updates its availability status bit S_<clock source> of the ME_GS register to '0'. The following clock sources are switched off at this step:

- the 16 MHz internal RC osc
- the FXOSC
- the PLL PHI_0

For the clock sources related to the system clock, this step is executed only after the System Clock Switching process has completed.

40.4.3.18 Flash Switch-Off

Based on the FLAON bit field of the ME_<current mode>_MC and ME_<target mode>_MC registers, if the flash is to be put in its low-power or power-down mode, the MC_ME requests the flash to enter the corresponding power mode and waits for the flash to acknowledge. The exact power mode status of the flash is updated in the S_FLA bit field of the ME_GS register. This step is executed only when the Processor Clocks Disable and System Clock Disable processes have completed.

40.4.3.19 Main Voltage Regulator Switch-Off

Based on the MVRON bit of the ME_current mode_MC and ME_target mode_MC registers, if the main voltage regulator is to be switched off, the MC_ME requests it to power down and clears the availability status bit S_MVR of the ME_GS register.

This step is required only during the entry of low-power modes like STOP0. This step is executed only after completing the following processes:

- Clock Sources Switch-Off
- Flash Switch-Off

If the target mode is STANDBY0, the main voltage regulator is not switched off by the MC_ME and the STANDBY0 request is asserted after the above processes have completed upon which the MC_PCU takes control of the main regulator. As the MC_PCU needs the 16 MHz int. RC osc., the 16 MHz internal RC oscillator remains active until all the STANDBY0 steps are executed by the MC_PCU after which it may be switched off depending on the IRCON bit of the ME_STANDBY0_MC register.

40.4.3.20 Current Mode Update

The current mode status bit field S_CURRENT_MODE of the ME_GS register is updated with the target mode bit field TARGET_MODE of the ME_MCTL register when :

- all the updated status bits in the ME_GS register match the configuration specified in the ME_<target mode>_MC register
- power sequences are done
- clock disable/enable process is finished
- processor low-power mode (stop) entry and exit processes are finished

Note

SAFE mode entry does not wait for the clock disable/enable process to finish. It only waits for the ME_GS.S_RC bit to be set. This is to ensure that the SAFE mode is entered as quickly as possible.

Software can monitor the mode transition status by reading the S_MTRANS bit of the ME_GS register. The mode transition latency can differ from one mode to another depending on the resources' availability before the new mode request and the target mode's requirements.

If a mode transition is taking longer to complete than is expected, the ME_DMTS register can indicate which process is still in progress.

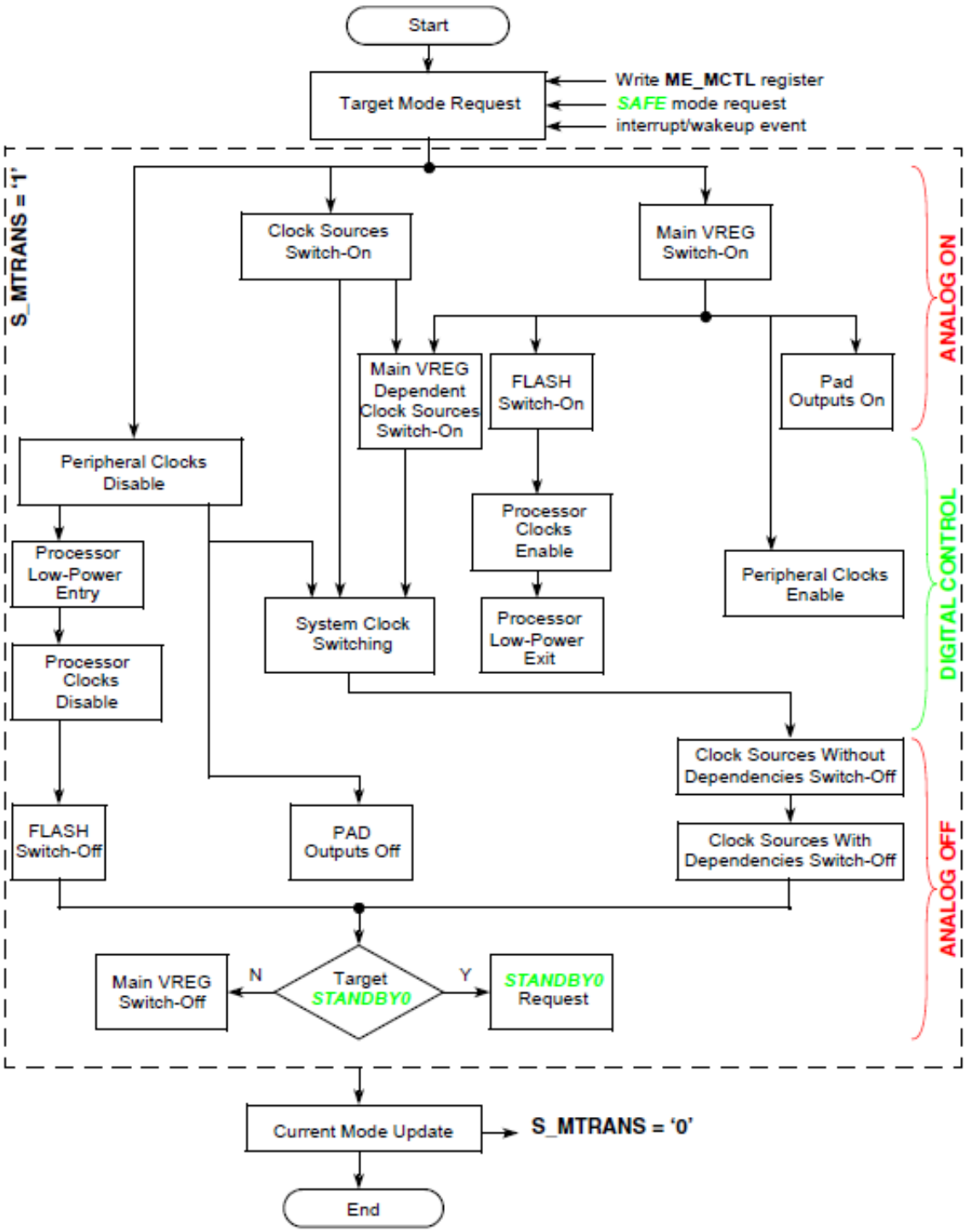


Figure 40-6. MC_ME Transition Diagram

40.4.3.21 Mode transition and progressive clock switching

The MC_ME initiates a progressive clock switching via MC_CGM whenever the **ME_<current mode>_MC.PWRLVL** field is different from **ME_<target mode>_MC.PWRLVL** bit field. Broadly, the mode transition process can be divided into three steps with respect to progressive cclock switching - RAMP DOWN, LOAD CHANGE and RAMP UP. One or more of these steps may not be needed based on the current and target mode configirations. The clock switching (from current system clock source to target system clock source) can happen before or after the LOAD CHANGE step.

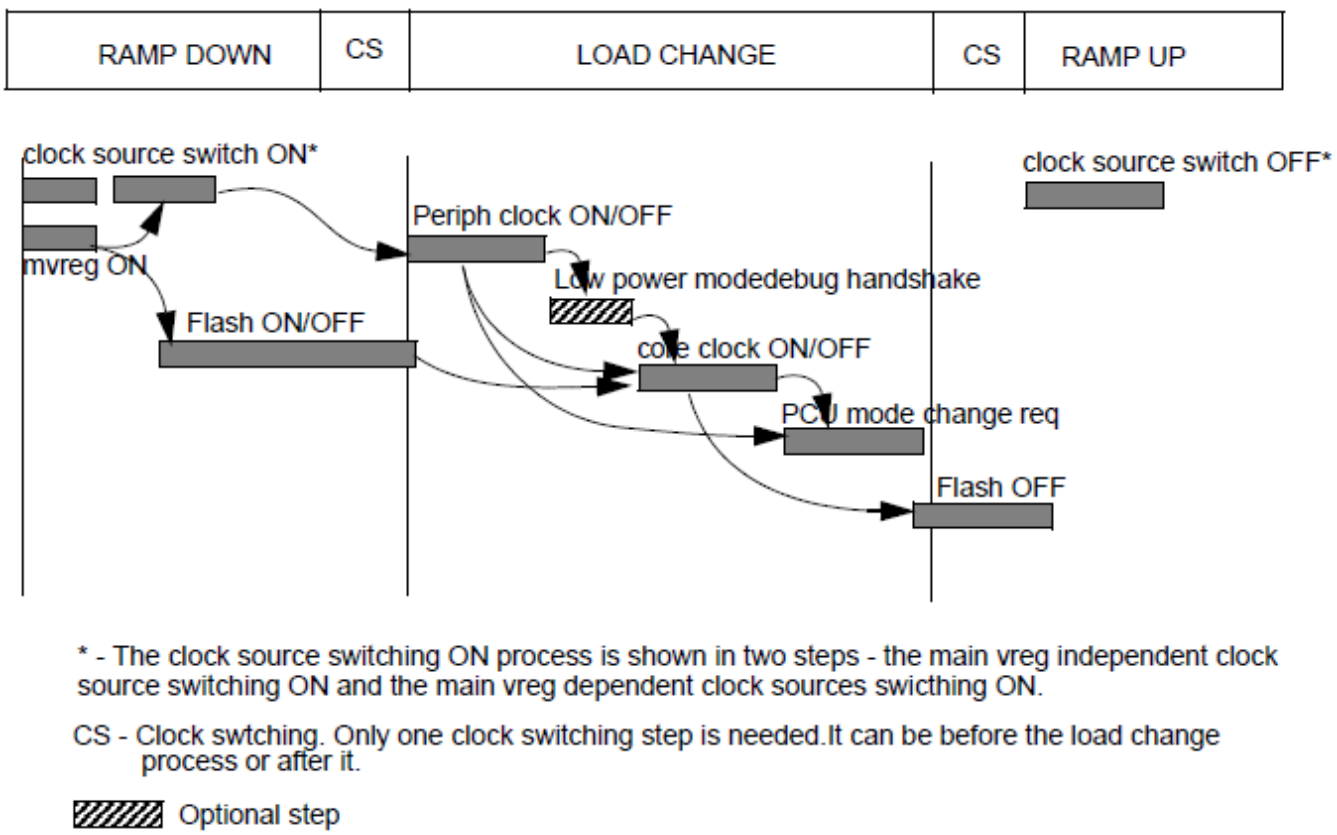


Figure 40-7. MC_ME mode transition process with PCS

The table below shows the different ways the mode transition process can complete.

Table 40-10. Mode transition steps

Pwrlvl	Source clock	Target clock	Mode transition steps
Same	IRC	Non-IRC	LoadChange -> ClockSwitch
	IRC	IRC	LoadChange
	Non-IRC	IRC	LoadChange -> ClockSwitch
	Non-IRC	Non-IRC	LoadChange -> ClockSwitch

Table continues on the next page...

Table 40-10. Mode transition steps (continued)

Pwrlvl	Source clock	Target clock	Mode transition steps
Different	IRC	Non-IRC	LoadChange -> ClockSwitch -> RampUp
	IRC	IRC	LoadChange
	Non-IRC	IRC	RampDown -> ClockSwitch -> LoadChange
	Non-IRC	Non-IRC	RampDown -> ClockSwitch -> LoadChange -> RampUp

40.4.4 Software Considerations for Preventing the Blocking of Mode Transitions

Mode transitions will not complete (i.e., mode transition is blocked) in certain conditions where

- resources are required in the target mode but are configured to be disabled in that target mode
- a low-power mode transition is attempted and there is a pending wakeup event
- a low-power mode transition is attempted and there is a pending interrupt

In these cases, ME_STATUS[S_MTRANS] never clears, ME_IS[I_MTC] never sets, and the invalid mode transition flags may not set.

While waiting for the mode transition to complete (e.g., polling ME_GS[S_MTRANS]), a timer can be used to detect if a mode transition takes too long. If the timer expires an error recovery function can be called.

To prevent mode transitions from being blocked, software must handle the following mode transition cases:

- **Enable required clock sources for the target mode's system clock selection.**

  Mode transitions can be blocked when a clock source to generate the system clock is not turned on in the target mode.

  Example for when FXOSC is the reference clock for PHI0:

  If the target mode is RUN1 and ME_RUN1_MC[SYSCLK] is PHI0, then both ME_RUN1_MC[FXOSCON] and ME_RUN1_MC[PHI0ON] must be set to '1'. If ME_RUN1_MC[FXOSCON] is '0', the mode transition is blocked.

  Software must ensure that all required clock sources for generating the target mode's system clock are turned on for that target mode configuration

- **Enable required clock sources for peripherals in the target mode.**

  Mode transitions can be blocked when an attempt is being made to stop a clock while one or more peripherals require this clock in the target mode.

  A given peripheral's clock is configured to be stopped in a target mode via peripheral configuration registers ME_RUN_PCx and ME_LP_PCx and the corresponding peripheral control register ME_PCTLx. In addition, some peripherals may require local disabling before stopping their clocks via the MC_ME. Local disabling is accomplished by setting the peripheral's MDIS bit to '1'.

  Modules requiring clocks being disabled before stopping them are:
  - CAN modules
  - display controllers (if included)

  Example:

  Assume a CAN module is using FXOSC. A mode transition to RUN1 is requested where MC_RUN1_MC[FXOSCON] is '0'. The mode transition will not complete.

  Software must disable the module whose clock source is to be turned off in the target mode.

- Disable or service any pending interrupts when transitioning to low-power modes.

  For transition into low power modes (STOP0, and STANDBY0), mode transitions can be blocked by a peripheral's pending interrupt.

  Pending interrupts are detected by the MC_ME when the peripheral's interrupt flag is set, and the peripheral's interrupt enable is set.

  NOTE

  The MSR[EE] and INTC_PSRx[PRI] registers have no effect on the MC_ME detecting a pending interrupt.

  Software must either
  - disable interrupt enable flags in peripherals that may have their flag set, or,
  - detect and process the interrupt by either
    - polling the peripheral flag and servicing it in software, or,
    - taking the interrupt vector (requires MSR[EE] = 1 and INTC_PSRx[PRI] >0 for that peripheral.

- **Service any pending wakeup events when transitioning to low power modes.**

  For transition into low-power modes (STOP0, and STANDBY0), mode transitions can be blocked by enabled wakeup events from the RTC, the API, or a wakeup pin transition.

  Example:

The API is set to a very short value and enabled as a wakeup event. After the first of two writes to the ME_MCTL register to request a mode transition, the API expires. The mode transition will not complete because the wakeup event requires servicing.

Software must either

- ensure wakeup events cannot occur before low-power mode entry completion, or,
- detect and process the wakeup events during mode transition by either
  - polling the flags of enabled wakeup events in the WISR register and servicing it in software, or,
  - enabling wakeup events to cause an interrupt and execute the interrupt service routine (ISR).

NOTE

If the ISR also invokes a mode transition on it's own, then any prior writes to the ME_MCTRL register for the low-power mode transition will not be valid.

40.4.5 Protection of Mode Configuration Registers

While programming the mode configuration registers ME_<mode>_MC, the following rules must be respected. Otherwise, the write operation is ignored and an invalid mode configuration interrupt may be generated.

- If the FIRC_CLK is selected as the system clock, FIRC must be on.
- If the FXOSC_CLK clock is selected as the system clock, FXOSC must be on.
- If the PLL_PHI0_CLK clock is selected as the system clock, PLL must be on.

Note

Software must ensure that clock sources with dependencies are switched on as needed. There is no automatic protection mechanism to check this in the MC_ME.

- Configuration "01" for the FLAON bit field is reserved.
- Configuration "10" for the FLAON bit field is reserved
- Configuration "00" for the FLAON bit field is reserved.
- MVREG must be on if any of the following is active:
  - the PLL PHI_0
  - FLASH
- System clock configurations marked as 'reserved' may not be selected.

40.4.6 Mode Transition Interrupts

The MC_ME provides interrupts for incorrectly configuring a mode, requesting an invalid mode transition, indicating a SAFE mode transition not due to a software request, and indicating when a mode transition has completed.

40.4.6.1 Invalid Mode Configuration Interrupt

Whenever a write operation is attempted to the ME_<mode>_MC registers violating the protection rules mentioned in the Protection of Mode Configuration Registers, the interrupt pending bit I_ICONF of the ME_IS register is set and an interrupt request is generated if the mask bit M_ICONF of the ME_IM register is '1'.

If an attempt to write to one of the ME_CADDRn registers is made after a mode change request has been made via the second write to the ME_MCTL register and before the completion of that mode transition as indicated by the S_MTRANS bit in the ME_GS register deasserting, the interrupt pending bit I_ICONF_CC of the ME_IS register is set and an iterrupt request is generated if the mask bit M_ICONF_CC of the ME_IM register is '1'.

40.4.6.2 Invalid Mode Transition Interrupt

The mode transition request is considered invalid under the following conditions:

- If the system is in the SAFE mode and the SAFE mode request from MC_RGM is active, and if the target mode requested is other than RESET or SAFE, then this new mode request is considered to be invalid, and the S_SEA bit of the ME_IMTS register is set.
- If the TARGET_MODE bit field of the ME_MCTL register is written with a value different from the specified mode values (i.e., a non-existing mode), an invalid mode transition event is generated. When such a non existing mode is requested, the S_NMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register. The following modes are defined as non-existing:
  - 1001
  - 1011
  - 1100
  - 1110

- If some of the chip modes are disabled as programmed in the ME_ME register, their respective configurations are considered reserved, and any access to the ME_MCTL register with those values results in an invalid mode transition request. When such a disabled mode is requested, the S_DMA bit of the ME_IMTS register is set. This condition is detected regardless of whether the proper key mechanism is followed while writing the ME_MCTL register.
- If the target mode is not a valid mode with respect to the current mode, the mode request illegal status bit S_MRI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If further new mode requests occur while a mode transition to a mode other than SAFE mode is in progress (the S_MTRANS bit of the ME_GS register is '1'), the mode transition illegal status bit S_MTI of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.
- If a new mode request occur while a mode transition to the SAFE mode is in progress (the S_MTRANS bit of the ME_GS register is ‘1’), the mode request ignored status bit S_MRIG of the ME_IMTS register is set. This condition is detected only when the proper key mechanism is followed while writing the ME_MCTL register. Otherwise, the write operation is ignored.

Note

As the causes of invalid mode transitions may overlap at the same time, the priority implemented for invalid mode transition status bits of the ME_IMTS register in the order from highest to lowest is S_SEA, S_NMA, S_DMA, S_MRI, S_MRIG, and S_MTI.

As an exception, the mode transition request is not considered as invalid under the following conditions:

- A new request is allowed to enter the RESET or SAFE mode irrespective of the mode transition status.

- As the exit of STOP0 modes depends on the interrupts of the system which can occur at any instant, these requests to return to RUN0…3 modes are always valid.
- In order to avoid any unwanted lockup of the chip modes, software can abort a mode transition by requesting the parent mode if, for example, the mode transition has not completed after a software determined 'reasonable' amount of time for whatever reason. The parent mode is the chip mode before a valid mode request was made.

Note

Depending on the cause of an incomplete mode transition, aborting by requesting the parent mode might not succeed. In this case, software must request a change to either SAFE or RESET mode, because the cause of the lockup is most likely a hardware fault.

- Self-transition requests (e.g., RUN0 → RUN0) are not considered as invalid even when the mode transition process is active (i.e., S_MTRANS is '1'). During the low-power mode exit process, if the system is not able to enter the respective RUN0…3 mode properly (i.e., all status bits of the ME_GS register match with configuration bits in the ME_<mode>_MC register), then software can only request the SAFE or RESET mode. It is not possible to request any other mode or to go back to the low-power mode again.

Whenever an invalid mode request is detected, the interrupt pending bit I_IMODE of the ME_IS register is set, and an interrupt request is generated if the mask bit M_IMODE of the ME_IM register is '1'.

40.4.6.3 SAFE Mode Transition Interrupt

Whenever the system enters the SAFE mode as a result of a SAFE mode request from the MC_RGM due to a hardware failure, the interrupt pending bit I_SAFE of the ME_IS register is set, and an interrupt is generated if the mask bit M_SAFE of ME_IM register is '1'.

The SAFE mode interrupt pending bit can be cleared only when the SAFE mode request is deasserted by the MC_RGM (see the MC_RGM chapter for details on how to clear a SAFE mode request). If the system is already in SAFE mode, any new SAFE mode request by the MC_RGM also sets the interrupt pending bit I_SAFE. However, the SAFE mode interrupt pending bit is not set when the SAFE mode is entered by a software request (i.e., programming of ME_MCTL register).

40.4.6.4 Mode Transition Complete interrupt

Whenever the system fully completes a mode transition (i.e., the S_MTRANS bit of ME_GS register transits from '1' to '0'), the interrupt pending bit I_MTC of the ME_IS register is set, and an interrupt request is generated if the mask bit M_MTC of the ME_IM register is '1'. The interrupt bit I_MTC is not set when entering low-power modes STOP0 in order to avoid the same event requesting the immediate exit of these low-power modes.

40.4.7 Peripheral Clock Gating

During all chip modes, each peripheral can be associated with a particular clock gating policy determined by two groups of peripheral configuration registers.

The run peripheral configuration registers ME_RUN_PC0…7 are chosen only during the software running modes DRUN, SAFE, and RUN0…3. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Run configuration selection for each peripheral is done by the RUN_CFG bit field of the ME_PCTL***n*** registers.

The low-power peripheral configuration registers ME_LP_PC0…7 are chosen only during the low-power modes STOP0 and STANDBY0. All configurations are programmable by software according to the needs of the application. Each configuration register contains a mode bit which determines whether or not a peripheral clock is to be gated. Low-power configuration selection for each peripheral is done by the LP_CFG bit field of the ME_PCTL***n*** registers.

Any modifications to the ME_RUN_PC0…7, ME_LP_PC0…7, and ME_PCTL***n*** registers do not affect the clock gating behavior until a new mode transition request is generated.

Whenever the chip enters a debug session during any mode, the following occurs for each peripheral:

- The clock is gated if the DBG_F bit of the associated ME_PCTL***n*** register is set. Otherwise, the peripheral clock gating status depends on the RUN_CFG and LP_CFG bits.

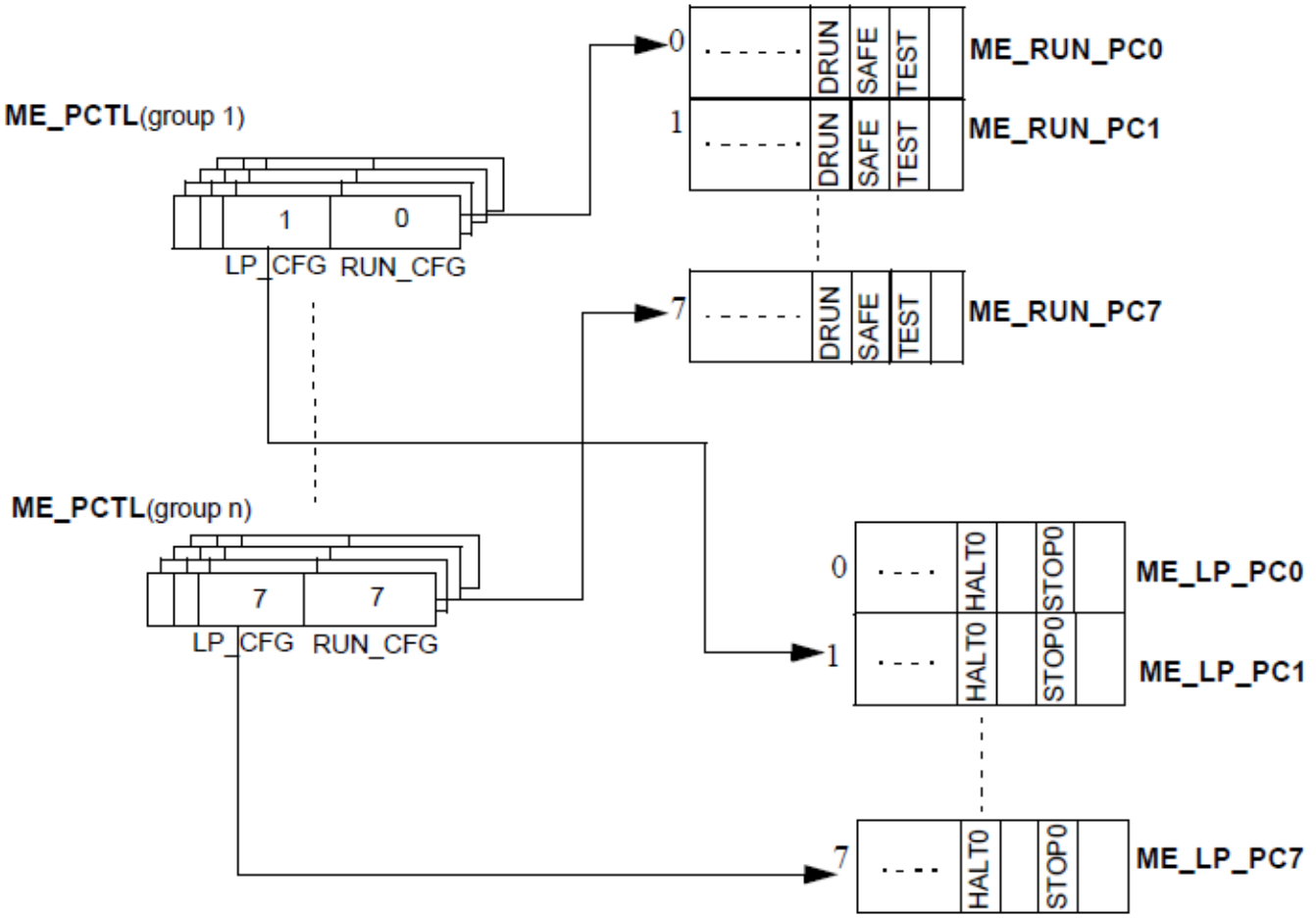


Figure 40-8. MC_ME Peripheral clock control

40.4.8 Initialization/Application Example

The following figure shows an example application flow for requesting a mode change and then waiting until the mode transition has completed.

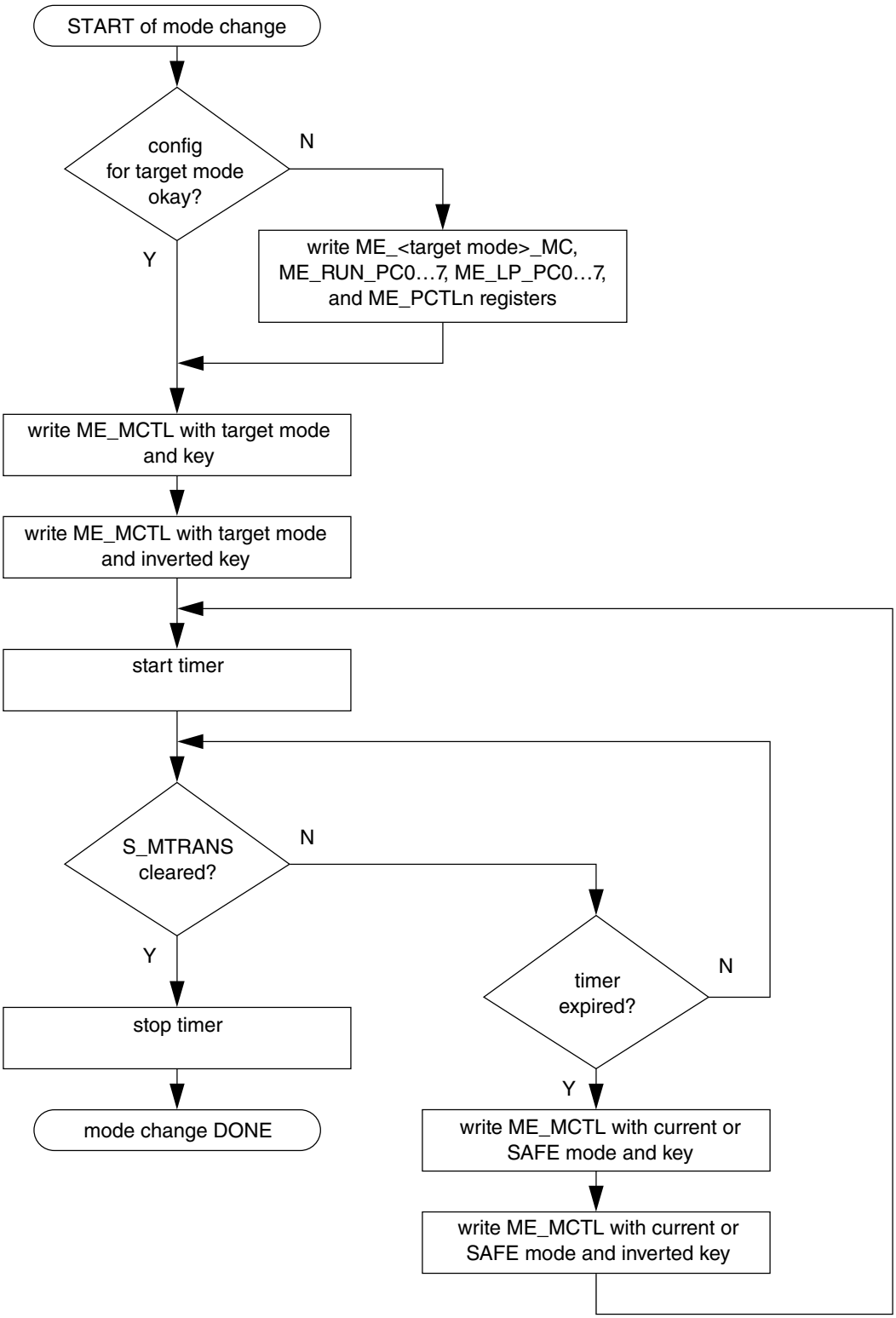


Figure 40-9. MC_ME Application Example Flow Diagram

40.4.8.1 Example 1: Minimal Code to Bring Up a PLL

The following minimal code brings up a PLL as the system clock and provides a clock to one peripheral (PIT-RTI). The peripheral itself would be initialized subsequently elsewhere. However, the peripherals registers cannot be written to for initialization unless there is a clock enabled to it. No additional cores are configured in this example.

Please note that the order of the code before the writes to the mode control register (MC_ME.MCTL) is not important.

```
void System_Clk_160mhz(void)
{
    /* F160 - max 160 MHz */
    MC_CGM.SC_DC0.B.DIV = 0;

    /* S80 - max 80 MHz */
    /* MC_CGM_SC_DC1[DIV] and MC_CGM_SC_DC5[DIV] must be equal at all times */
    MC_CGM.SC_DC1.B.DIV = 1;

    /* FS80 - max 80 MHz */
    /* MC_CGM_SC_DC1[DIV] and MC_CGM_SC_DC5[DIV] must be equal at all times */
    MC_CGM.SC_DC5.R = MC_CGM.SC_DC1.R;  /* 80 MHz max */

    /* S40 - max 40 MHz */
    MC_CGM.SC_DC2.B.DIV = 3;

    /* F40 - max 40 MHz */
    MC_CGM.SC_DC3.B.DIV = 3;

    /* F80 - max 80 MHz */
    MC_CGM.SC_DC4.B.DIV = 1;

    /* enable all modes, enable all peripherals */
    MC_ME.ME.R        = 0x000005FF;
    MC_ME.RUN_PC[0].R = 0x000000FE;

    /* Connect XOSC to PLL */
    MC_CGM.AC5_SC.B.SELCTL = 1;

    /* Configure PLL0 Dividers - 160MHz from 40Mhx XOSC */
    PLLDIG.PLLDV.B.PREDIV  = 2;
    PLLDIG.PLLDV.B.MFD     = 32;
    PLLDIG.PLLDV.B.RFDPHI  = 1;

    PLLDIG.PLLCAL3.R       = 0x09C3C000;
    PLLDIG.PLLFD.B.SMDEN   = 1;       /* Sigma delta modulation enabled*/
    /* switch to PLL */
    MC_ME.DRUN_MC.R  = 0x00130172;    /* FLASH in normal mode, PLLON, FXOSC ON, Use
PLL_PHI_0 */

    MC_ME.MCTL.R     = 0x30005AF0;    /* DRUN target mode + KEY */
    MC_ME.MCTL.R     = 0x3000A50F;    /* DRUN target mode + KEY Inverted */

    while(MC_ME.GS.B.S_MTRANS == 1); /* Wait for mode transition complete */
}
```

40.4.8.2 Example 2: Code to Enter Low-Power STOP0 Mode

The following code illustrates how STOP0 mode is entered. Upon STOP0 exit from a wakeup event, code simply continues to execute. If an interrupt caused STOP0 mode exit, the interrupt handler code would execute. Depending on the interrupt handler task, it may be necessary to ensure that the chip is not in SAFE mode with code in the interrupt handler itself.

```
ME.MCTL.R = 0xA0005AF0; /* Enter STOP0 mode and key */

ME.MCTL.R = 0xA000A50F; /* Enter STOP0 mode and inverted key */

while (ME.GS.B.S_MTRANS) {} /* Wait for STOP0 mode transition to complete */

/* STOP HERE FOR STOP0 MODE! ON STOP0 MODE EXIT, CODE CONTINUES HERE:*/

while(ME.GS.B.S_CURRENTMODE != x) {} /* Verify RUNx is the current mode */
```

Upon low-power mode exit, the current mode should be verified to make sure it is the RUNx mode that was in use during the low-power mode entry because a SAFE mode transition (caused by a hardware failure) may have occurred during the low-power mode exit.

If the low-power mode exit was caused by an interrupt, the interrupt handler may, depending on the application and handler, need to verify the current mode is not SAFE mode, since the check at the end of the code snippet above may not be executed before the jump to the interrupt vector.

Chapter 41
General Programmable Registers (GPR)

41.1 Overview

This module consists of registers which are used to control miscellaneous tasks in the device and capture the event status.

41.2 Memory map and register definition

GPR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	GPR Control Register (GPR_CTL)	32	R/W	See section	41.2.1/1212
4	GPR Status Register (GPR_STAT)	32	w1c	0000_0002h	41.2.2/1215
10	GPR Control Register1 (GPR_CTL1)	32	R/W	0000_0000h	41.2.3/1216
14	GPR SLEEP Register (GPR_SLEEP)	32	R/W	See section	41.2.4/1216

41.2.1 GPR Control Register (GPR_CTL)

This register is used for controlling miscellaneous tasks in the device.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	GPR_HSM_RAM_FAST_INIT_BYPASS	Reserved		EXTREGC	DEST_ESC_CTRL2	DEST_ESC_CTRL1	DEST_ESC_CTRL0	Reserved					CMP2_STDBY	CMP1_STDBY	CMP0_STDBY	CLKOUT1_STDBY
W																
Reset	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LPU_STOP_ACK_BYPASS	Reserved	Reserved	RTC_SXOSC_FREE_RUNNING	CMP_ACT_SAM	CMP_SAMP_SEL	Reserved		Reserved		FR_BS		ENET_TX_BS		ENET_RX_BS	
W																
Reset	0	0	*	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:
- Reserved field: This bit should never be set.

GPR_CTL field descriptions

Field	Description
0 GPR_HSM_RAM_FAST_INIT_BYPASS	HSM RAM FAST initialization is bypassed on low power mode exit if this bit is set . 0 HSM RAM FAST initialization is not bypassed 1 HSM RAM FAST initialization is bypassed
1–2 Reserved	This field is reserved.
3 EXTREGC	This bit enables the external regulator control multiplexed function on PG[6] The PG[6] pin is configured as an output and is an indicator of a partial power-gated state of the SOC. The signal can be used to control and external regulator on the VDD_LV supply. This bit will reset only on PORST. 0 External regulator control functionality on PG[6] is disabled 1 External regulator control functionality on PG[6] is enabled

Table continues on the next page...

GPR_CTL field descriptions (continued)

Field	Description
4 DEST_ESC_ CTRL2	Enables the escalation of destructive reset source: SWT0 reset request. This bit will be reset on POR. 0 destructive reset escalation of SWT0 reset request is disabled 1 destructive reset escalation of SWT0 reset request is enabled (default)
5 DEST_ESC_ CTRL1	Enables the escalation of destructive reset source: functional reset escalation. This bit will be reset on POR. 0 destructive reset escalation of functional reset escalation is disabled 1 destructive reset escalation of functional reset escalation is enabled (default)
6 DEST_ESC_ CTRL0	Enables the escalation of destructive reset source: SSCM secured reset. This bit will be reset on POR. 0 destructive reset escalation of SSCM secured reset is disabled 1 destructive reset escalation of SSCM secured reset is enabled (default)
7–11 Reserved	This field is reserved.
12 CMP2_STDBY	To allow CMP2_O to be output to pad even after standby exit 0 The CMP2_O will not show the output after standby modes exit 1 The CMP2_O will show the the output after standby modes exit
13 CMP1_STDBY	To allow CMP1_O to be output to pad even after standby exit 0 The CMP1_O will not show the output after standby modes exit 1 The CMP1_O will show the the output after standby modes exit
14 CMP0_STDBY	To allow CMP0_O to be output to pad even after standby exit 0 The CMP0_O will not show the output after standby modes exit 1 The CMP0_O will show the the output after standby modes exit
15 CLKOUT1_ STDBY	To allow CLKOUT1 to be output to pads even during standby mode **NOTE:** If this bit is set before entering STANDBY/LPU_STANDBY mode, and SIUL MSCR is configured to enable CLKOUT1 on SIUL pad, then upon exit from STANDBY/LPU_STANDBY mode, CLKOUT1 remains selected even if SIUL MSCR configurations are cleared. In order to configure the CLKOUT1 pad for any other function via SIUL, this bit needs to be cleared first. 0 CLKOUT1 will not be output to pads during standby mode 1 CLKOUT1 will be output to pads even during standby mode
16 LPU_STOP_ ACK_BYPASS	Bypass STOP acknowledge handshake with LPU domain modules (which are present in LPU_MDIS register), during LPU_STOP and LPU_SLEEP. **NOTE:** Modules should be IDLE before entering LPU_STOP/SLEEP, that is, no outstanding transactions or interrupts should be pending. Bypassing STOP ACK will enable entry in LPU_STOP/SLEEP irrespective of interrupt generated during the mode transition and then the same interrupt will cause mode exit. If STOP Handshake is not bypassed, mode transition may hang as modules will not acknowledge stop due to outstanding interrupt during the mode transition. 0 Enable STOP ACK handshake(Default) 1 Bypass STOP ACK handshake
17 Reserved	This field is reserved.
18 Reserved	This field is reserved.

Table continues on the next page...

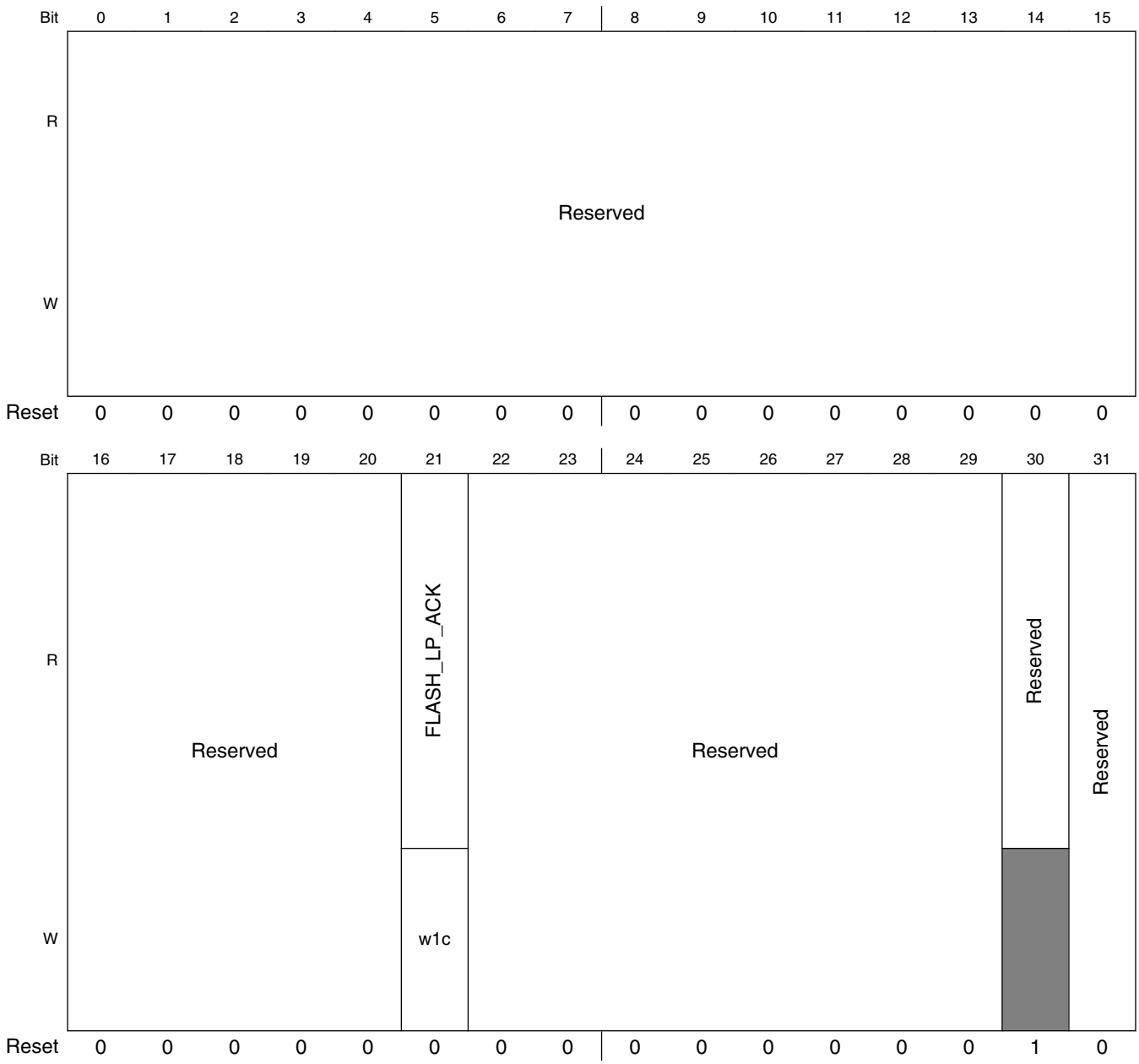
GPR_CTL field descriptions (continued)

Field	Description
19 RTC_SXOSC_ FREE_RUNNING	Enables seamless RTC operation across functional and destructive reset in the specific case when RTC is running on SXOSC. **NOTE** This Bit should be set after SXOSC is enabled by the software. 0 RTC will not work on SXOSC seamlessly across functional and destructive reset. 1 RTC will work on SXOSC seamlessly across functional and destructive reset.
20 CMP_ACT_SAM	Actual sample to the CMP 0 Sample of all three CMP = 0 when GPR_CTL [CMP_SAMP_SEL] = 1 1 Sample of all three CMP = 1 when GPR_CTL [CMP_SAMP_SEL] = 1
21 CMP_SAMP_ SEL	Sample select to the CMP 0 RTC.CLKOUT goes to "sample" of all three CMP 1 GPR_CTL [CMP_ACT_SAM] goes to "sample" of all three CMP
22–23 Reserved	This field is reserved.
24–25 Reserved	This field is reserved.
26–27 FR_BS	FlexRay byte swapper: Swaps the AHB write/read data at the boundary of FlexRay master 00 No swap 01 8 bytes swap 10 Reserved 11 Reserved
28–29 ENET_TX_BS	ENET TX byte swapper: Swaps the AHB write/read data at the TX boundary of ENET master 00 No swap 01 8 bytes swap 10 Reserved 11 Reserved
30–31 ENET_RX_BS	ENET RX byte swapper: Swaps the AHB write/read data at the RX boundary of ENET master 00 No swap 01 8 bytes swap 10 Reserved 11 Reserved

41.2.2 GPR Status Register (GPR_STAT)

This register is used to capture the flash low power acknowledge status.

Address: 0h base + 4h offset = 4h



GPR_STAT field descriptions

Field	Description
0–20 Reserved	This field is reserved.

Table continues on the next page...

GPR_STAT field descriptions (continued)

Field	Description
21 FLASH_LP_ACK	FLASH low power acknowledge status 0 The flash low power acknowledge status is disabled 1 The flash low power acknowledge status is enabled
22–29 Reserved	This field is reserved.
30 Reserved	This field is reserved.
31 Reserved	This field is reserved.

41.2.3 GPR Control Register1 (GPR_CTL1)

This register defines the reset value of the Interrupt vector Prefix register of all the twocores z4and z2. When exiting Standby mode with boot vector pointing to RAM location, IVPR of the cores can be directed to RAM address location..

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								IVPR																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPR_CTL1 field descriptions

Field	Description
0–7 Reserved	This field is reserved.
8–31 IVPR	Interrupt Vector Prefix Register Programming This field is used to define the base location of the interrupt vector table

41.2.4 GPR SLEEP Register (GPR_SLEEP)

This register defines the sleep status of SRAM memory blocks

NOTE

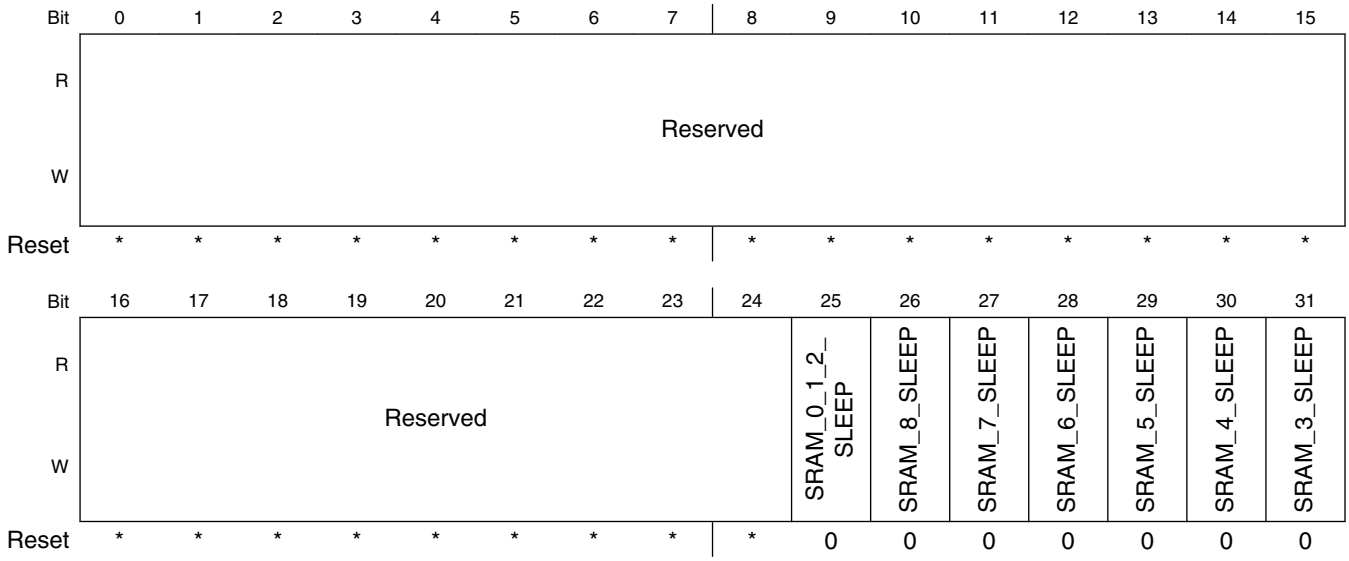
See RAM memory map for bit details.

NOTE

An attempt to SRAM Read with Sleep Bit set will result in incorrect read with ECC error, hence before attempting an

SRAM Read Sleep bit for the corresponding SRAM should be cleared.

Address: 0h base + 14h offset = 14h



* Notes:
- Reserved field: This bit should never be set.

GPR_SLEEP field descriptions

Field	Description
0–24 Reserved	This field is reserved.
25 SRAM_0_1_2_SLEEP	Enables the sleep mode for SRAM memory block 0, 1, 2. 0 Sleep mode disabled 1 Sleep mode enabled
26 SRAM_8_SLEEP	Enables the sleep mode for SRAM memory block 8. 0 Sleep mode disabled 1 Sleep mode enabled
27 SRAM_7_SLEEP	Enables the sleep mode for SRAM memory block 7. 0 Sleep mode disabled 1 Sleep mode enabled
28 SRAM_6_SLEEP	Enables the sleep mode for SRAM memory block 6. 0 Sleep mode disabled 1 Sleep mode enabled
29 SRAM_5_SLEEP	Enables the sleep mode for SRAM memory block 5. 0 Sleep mode disabled 1 Sleep mode enabled

Table continues on the next page...

GPR_SLEEP field descriptions (continued)

Field	Description
30 SRAM_4_SLEEP	Enables the sleep mode for SRAM memory block 4. 0 Sleep mode disabled 1 Sleep mode enabled
31 SRAM_3_SLEEP	Enables the sleep mode for SRAM memory block 3. 0 Sleep mode disabled 1 Sleep mode enabled

Chapter 42
Deserial Serial Peripheral Interface (DSPI)

42.1 Chip-specific DSPI information

42.1.1 DSPI and SPI configuration

This section describes the configuration of these modules specific to the MPC5746C.

42.1.2 DSPI configuration

The device contains four DSPI[0:3] modules. DSPI_2 and DSPI_3 are targeted for faster frequency.

Table 42-1. Full Featured DSPI[1:0] configuration

Feature		Available
Extended (32-bit) SPI Mode Support		Yes
CSI and DSI mode support.		Yes
64 bit DSI Mode Support		Yes
TX FIFO		4
RX FIFO		4
CTAR registers (transfer attribute configuration)		6
TSB Mode		Yes
PCS Signals	DSPI0	DSPI0_PCS[0:5]
	DSPI1	DSPI1_PCS[0:4]

Table 42-2. Full Featured DSPI[3:2] configuration

Feature	Available
Extended (32-bit) SPI Mode Support	Yes
CSI and DSI mode support.	Yes

Table continues on the next page...

Table 42-2. Full Featured DSPI[3:2] configuration (continued)

64 bit DSI Mode Support		No
TX FIFO		4
RX FIFO		4
CTAR registers (transfer attribute configuration)		6
TSB Mode		Yes
PCS Signals	DSPI2	DSPI2_PCS[0:3]
	DSPI3	DSPI3_PCS[0:1]

42.1.3 SPI configuration

The device contains 4 SPI[0:3] modules. SPI_1and SPI_2 are targeted for faster frequency.

Table 42-3. SPI[0:3] Configuration

Feature		Available
32-bit support		Yes
Extended (32-bit) SPI Mode Support		Yes
CSI and DSI mode support.		No
64 bit DSI Mode Support		No
TX FIFO		4
RX FIFO		4
CTAR registers (transfer attribute configuration)		6
TSB Mode		No
PCS Signals	SPI0	SPI0_PCS[0:3]
	SPI1	SPI1_PCS[0:2]
	SPI2	SPI2_PCS[0:3]
	SPI3	SPI3_PCS[0:3]

42.1.4 DSPI and SPI registers

The following table compares various registers available in DSPI and SPI modules in this device.

DSPI registers	SPI registers
Module Configuration Register (MCR)	Module Configuration Register (MCR)
Transfer Count Register (TCR)	Transfer Count Register (TCR)

Table continues on the next page...

DSPI registers	SPI registers
Clock and Transfer Attributes Register (In Master Mode) (CTARn)	Clock and Transfer Attributes Register (In Master Mode) (CTARn)
Clock and Transfer Attributes Register (In Slave Mode) (CTARn_SLAVE)	Clock and Transfer Attributes Register (In Slave Mode) (CTARn_SLAVE)
Status Register (SR)	Status Register (SR)
DMA/Interrupt Request Select and Enable Register (RSER)	DMA/Interrupt Request Select and Enable Register (RSER)
PUSH TX FIFO Register In Master Mode	PUSH TX FIFO Register In Master Mode
PUSH TX FIFO Register In Slave Mode (PUSHR_SLAVE)	PUSH TX FIFO Register In Slave Mode (PUSHR_SLAVE)
POP RX FIFO Register (POPR)	POP RX FIFO Register (POPR)
Transmit FIFO Registers (TXFRn)	Transmit FIFO Registers (TXFRn)
Receive FIFO Registers (RXFRn)	Receive FIFO Registers (RXFRn)
DSI Configuration Register 0 (DSICR0)	
DSI Serialization Data Register 0 (SDR0)	
DSI Alternate Serialization Data Register 0 (ASDR0)	
DSI Transmit Comparison Register 0 (COMPR0)	
DSI Deserialization Data Register 0 (DDR0)	
DSI Configuration Register 1 (DSICR1)	
DSI Serialization Source Select Register 0 (SSR0)	
DSI Parallel Input Select Register 0 (PISR0)	
DSI Parallel Input Select Register 1 (PISR1)	
DSI Parallel Input Select Register 2 (PISR2)	
DSI Parallel Input Select Register 3 (PISR3)	
DSI Deserialized Data Interrupt Mask Register 0 (DIMR0)	
DSI Deserialized Data Polarity Interrupt Register 0 (DPIR0)	
DSI Serialization Data Register 1 (SDR1)	
DSI Alternate Serialization Data Register 1 (ASDR1)	
DSI Transmit Comparison Register 1 (COMPR1)	
DSI Deserialization Data Register 1 (DDR1)	
DSI Serialization Source Select Register 1 (SSR1)	
DSI Parallel Input Select Register 4 (PISR4)	
DSI Parallel Input Select Register 5 (PISR5)	
DSI Parallel Input Select Register 6 (PISR6)	
DSI Parallel Input Select Register 7 (PISR7)	
DSI Deserialized Data Interrupt Mask Register 1 (DIMR1)	
DSI Deserialized Data Polarity Interrupt Register 1 (DPIR1)	
Clock and Transfer Attributes Register Extended (CTAREn)	Clock and Transfer Attributes Register Extended (CTAREn)
Status Register Extended (SREX)	Status Register Extended (SREX)
Trigger Register (TRIG)	

NOTE

Before re-configuring DSPI, it should be disabled by setting its MDIS bit. Changing the configuration of DSPI while the module's clock is still running should be avoided.

42.1.5 DSPI's stop-stop ack protocol

1. If DSPI is disabled, it acknowledges the stop request immediately.
2. In master mode, DSPI asserts the acknowledgement for the stop request on frame boundary of the frame being transmitted/received. Additionally, if no frame is being transmitted/received DSPI immediately asserts the acknowledgement.
3. In slave mode, DSPI asserts the acknowledgement for stop request when it's slave select is high (inactive value).

NOTE

In both 2 and 3, DSPI checks for pending Interrupt/DMA requests before asserting the acknowledgement.

42.2 Introduction

The deserial serial peripheral interface (DSPI) module provides a synchronous serial bus for communication between a chip and an external peripheral device. The module supports chip pin count reduction through serialization and deserialization of chip internal signals transmitted over the DSPI serial link.

42.2.1 Block Diagram

The block diagram of this module is as follows:

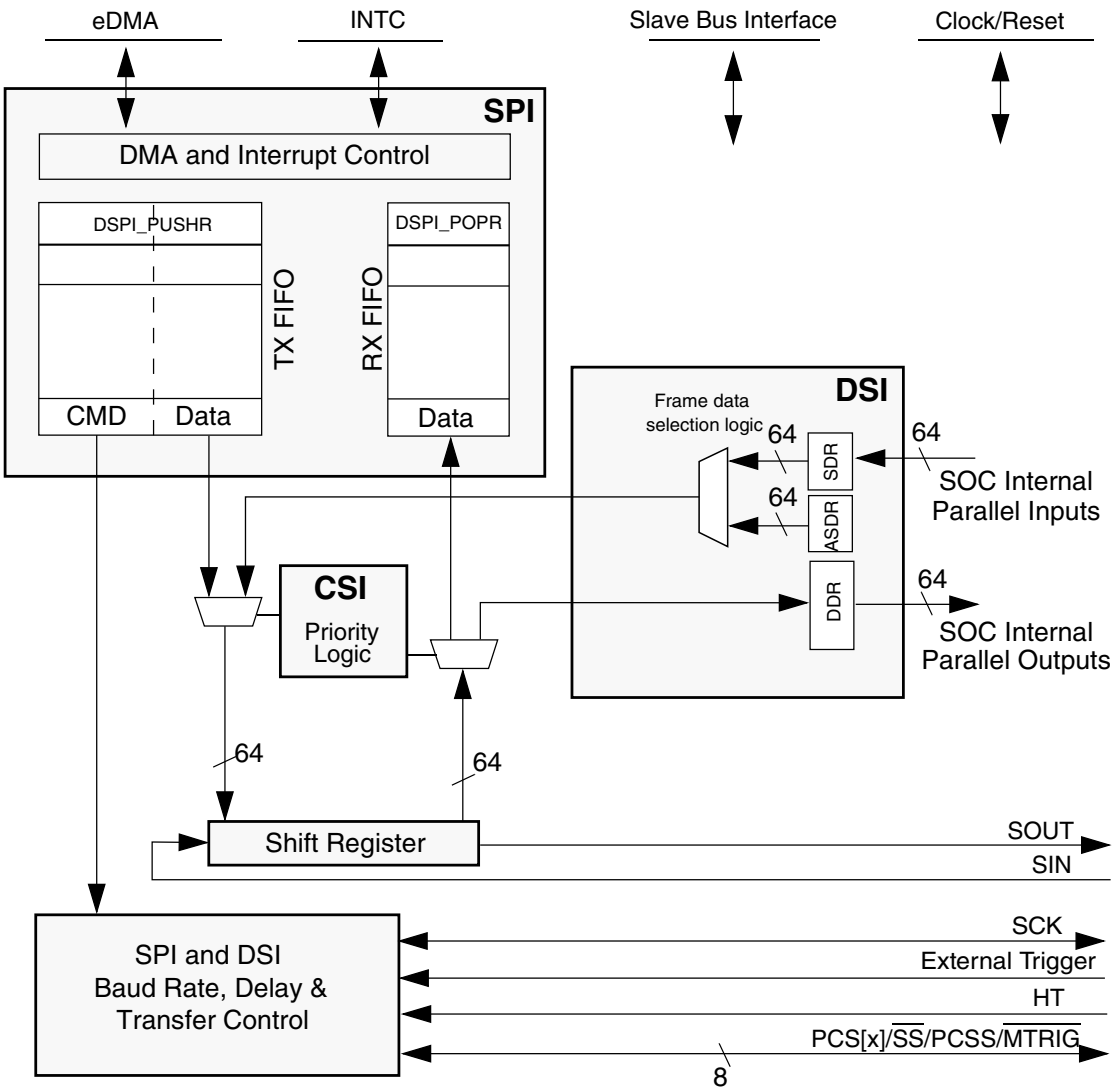


Figure 42-1. DSPI Block Diagram

42.2.2 Features

The module supports the following features:

- Full-duplex, three-wire synchronous transfers
- Master mode
- Slave mode
- Data streaming operation in Slave mode with continuous slave selection
- Buffered transmit operation using the transmit first in first out (TX FIFO) with depth of 4 entries

- Support for 8/16-bit accesses to the PUSH TX FIFO Register Data Field
- Buffered receive operation using the receive FIFO (RX FIFO) with depth of 4 entries
- Asynchronous clocking scheme for Register and Protocol Interfaces
- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues
- Visibility into TX and RX FIFOs for ease of debugging
- Programmable transfer attributes on a per-frame basis:
  - six transfer attribute registers
  - six extended transfer attribute registers
  - Serial clock (SCK) with programmable polarity and phase
  - Various programmable delays
  - Programmable serial frame size: 4 to 64 bits
    - SPI frames longer than 32 bits can be supported using the continuous selection format.
  - Continuously held chip select capability
  - Parity control
- 6 peripheral chip selects (PCSes), expandable to 64 with external demultiplexer
- Deglitching support for up to 32 peripheral chip selects (PCSes) with external demultiplexer
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
  - TX FIFO is not full (TFFF)
  - RX FIFO is not empty (RFDF)
  - CMD FIFO is not full (CMDFFF)
- Interrupt conditions:
  - End of Queue reached (EOQF)
  - TX FIFO is not full (TFFF)
  - CMD FIFO is not full (CMDFFF)
  - Transfer of current frame complete (TCF)
  - Transfers due from current command frame complete (CMDTCF)

    - Transfer of current SPI frame complete (SPITCF)
    - Attempt to transmit with an empty Transmit FIFO (TFUF)
    - RX FIFO is not empty (RFDF)
    - Frame received while Receive FIFO is full (RFOF)
    - SPI Parity Error (SPEF)
    - Data present in TX FIFO while CMD FIFO is empty (TFIWF)
- Global interrupt request line
- Modified SPI transfer formats for communication with slower peripheral devices
- Power-saving architectural features:
    - Support for Stop mode

The DSPI also supports pin reduction through serialization and deserialization if enabled for the module.

- Two sources of serialized data:
    - The module memory-mapped register
    - Parallel Input signals
    - Programmable selection of source data on bit basis
- Deserialized data is provided as Parallel Output signals and as bits in a memory-mapped register
- Interrupt conditions:
    - Deserialized data matches pre-programmed pattern (DDIF)
    - Transfer of current DSI frame complete (DSITCF)
    - DSI Parity Error (DPEF)
- DMA Request support for following conditions:
    - Deserialized data matches pre-programmed pattern (DDIF)
- Transfer initiation conditions:
    - Continuous
    - Edge sensitive hardware trigger
    - Change in data

- Support for parallel and serial chaining of several modules inside the chip
- Pin serialization/deserialization with interleaved SPI frames for control and diagnostics
- Continuous serial communications clock

DSPI supports a combined SPI and DSI modes of operation known as the Combined Serial Interface (CSI). It also supports the downstream Micro Second Channel in the Timed Serial Bus (TSB) configuration or Interleaved Frames Configuration; configurable by software.

Both TSB and Interleaved TSB modes have the following common features:

- Transmission of frames is done at frame boundaries
- Frames from SPI and DSI are identifiable by a bit transmitted at the start of each frame
- Separate interrupts for frame completion from SPI and DSI

42.2.3 Interface configurations

42.2.3.1 SPI configuration

The Serial Peripheral Interface (SPI) configuration allows the module to send and receive serial data. This configuration allows the module to operate as a basic SPI block with internal FIFOs supporting external queue operation. Transmitted data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the Receive FIFO and write transmit data to the Transmit FIFO.

For queued operations, the SPI queues can reside in system RAM, external to the module. Data transfers between the queues and the module FIFOs are accomplished by a DMA controller or host CPU. The following figure shows a system example with DMA, DSPI, and external queues in system RAM.

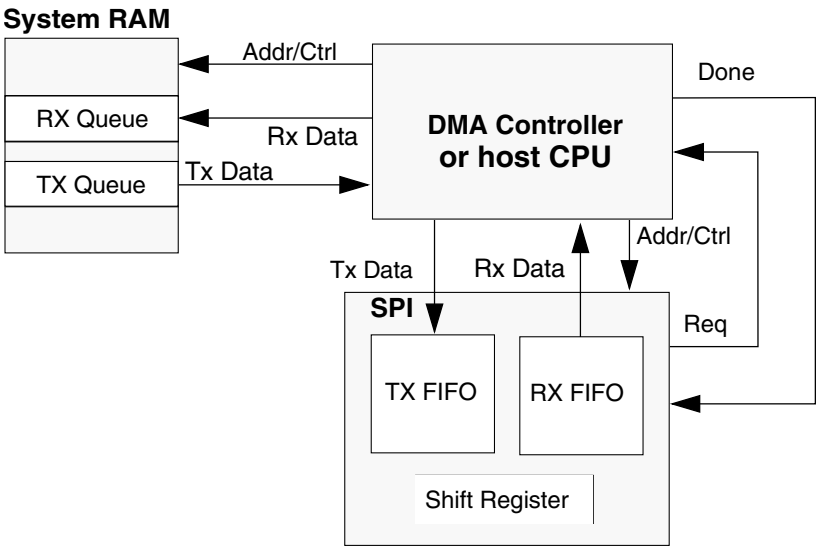


Figure 42-2. SPI with queues and DMA

42.2.3.2 DSI configuration

In the Deserial Serial Interface (DSI) configuration, DSPI serializes up to 64 parallel input signals or register bits. DSPI also deserializes the received data to parallel output signals or to a memory-mapped register. DSPI transfers the data using an SPI-like protocol.

Timed Serial Bus (TSB) mode provides the MicroSecond Channel (MSC) downstream support, serializing from 4 to 64 parallel input signals or register bits. For more information on TSB mode, see Timed Serial Bus (TSB).

42.2.3.3 CSI configuration

The Combined Serial Interface (CSI) configuration supports further configuration as follows:

- Normal: A combination of the SPI and DSI configurations. The DSPI interleaves DSI data frames with SPI data frames. Interleaving is done on the frame boundaries.
- TSB: Transmission of SPI data has higher priority than DSI data.
- Interleaved TSB (ITSB): The frames from SPI and DSI are interleaved without priority. On every trigger, frames from DSI are sent when there are no frames in the SPI or the previous transmission was a frame from SPI. For more information, see Interleaved TSB (ITSB) Mode.

42.2.4 Modes of Operation

The module supports the following modes of operation that can be divided into two categories:

- Module-specific modes:
  - Master mode
  - Slave mode
  - Module Disable mode
- Chip-specific modes:
  - External Stop mode
  - Debug mode

The module enters module-specific modes when the host writes a module register. The chip-specific modes are controlled by signals external to the module. The chip-specific modes are modes that a chip may enter in parallel to the block-specific modes.

42.2.4.1 Master Mode

Master mode allows the module to initiate and control serial communication. In this mode, these signals are controlled by the module and configured as outputs:

- SCK
- PCS[*x*]

42.2.4.2 Slave Mode

Slave mode allows the module to communicate with SPI/DSI bus masters. In this mode, the module responds to externally controlled serial transfers. The SCK signal and the PCS[0]/$\overline{SS}$ signals are configured as inputs and driven by an SPI bus master.

42.2.4.3 Module Disable Mode

The Module Disable mode can be used for chip power management. The clock to the non-memory mapped logic in the module can be stopped while in the Module Disable mode.

42.2.4.4 External Stop Mode

External Stop mode is used for chip power management. The module supports the Peripheral Bus Stop mode mechanism. When a request is made to enter External Stop mode, it acknowledges the request and completes the transfer that is in progress. When the module reaches the frame boundary, it signals that the protocol clock to the module may be shut off.

42.2.4.5 Debug Mode

Debug mode is used for system development and debugging. The MCR[FRZ] bit controls module behavior in the Debug mode:

- If the bit is set, the module stops all serial transfers, when the chip is in debug mode.
- If the bit is cleared, the chip debug mode has no effect on the module.

42.3 Module signal descriptions

This table describes the signals on the boundary of the module that may connect off chip (in alphabetical order).

Table 42-4. Module signal descriptions

Signal	Master mode	Slave mode	I/O
HT	Hardware Trigger	Hardware Trigger	I
PCS0/$\overline{SS}$	Peripheral Chip Select 0 (O)	Slave Select (I)	I/O
PCS[1:3]	Peripheral Chip Selects 1–3	(Unused)	O
PCS4/ $\overline{MTRIG}$	Peripheral Chip Select 4	Master Trigger	O
PCS5/ $\overline{PCSS}$	Peripheral Chip Select 5 /Peripheral Chip Select Strobe	(Unused)	O
SCK	Serial Clock (O)	Serial Clock (I)	I/O
SIN	Serial Data In	Serial Data In	I
SOUT	Serial Data Out	Serial Data Out	O

42.3.1 HT—Hardware Trigger

Master mode: Hardware Trigger (I)—Honored only when the module is configured for the DSI or CSI configuration (MCR[DCONF]), the multiple transfer operation is enabled (DSICR0[MTOE]), hardware-trigger reception is enabled (DSICR0[TRRE] or

TRIG[TRRE]), and the module is in the Running state (SR[TXRXS]). Receives hardware triggers that initiate data transfers. DSICR0[TPOL] specifies whether the module uses rising or falling edges of HT as hardware triggers.

Slave mode: Hardware Trigger (I)—Honored only when the module is configured for the DSI or CSI configuration (MCR[DCONF]), the multiple transfer operation is enabled (DSICR0[MTOE]), hardware-trigger reception is enabled (DSICR0[TRRE] or TRIG[TRRE]), and the module is in the Running state (SR[TXRXS]). Receives hardware triggers that initiate the assertion of $\overline{\text{MTRIG}}$. DSICR0[TPOL] specifies whether the module uses rising or falling edges of HT as hardware triggers.

42.3.2 PCS0/$\overline{\text{SS}}$—Peripheral Chip Select/Slave Select

Master mode: Peripheral Chip Select 0 (O)—Selects an SPI slave to receive data transmitted from the module.

Slave mode: Slave Select (I)—Selects the module to receive data transmitted from an SPI master.

NOTE

Do not tie the DSPI slave select pin to ground. Otherwise, DSPI cannot function properly.

42.3.3 PCS1–PCS3—Peripheral Chip Selects 1–3

Master mode: Peripheral Chip Selects 1–3 (O)—Select an SPI slave to receive data transmitted by the module.

Slave mode: Unused

42.3.4 PCS4/$\overline{\text{MTRIG}}$—Peripheral Chip Select 4/Master Trigger

Master mode: Peripheral Chip Select 4 (O)—Selects an SPI slave to receive data transmitted by the module.

Slave mode: Master Trigger (O)—Used only when the multiple transfer operation is enabled (DSICR0[MTOE]). Indicates via a pulse that a change in data to be serialized has occurred. The pulse is 4 protocol clock cycles in duration.

42.3.5 PCS5/$\overline{\text{PCSS}}$—Peripheral Chip Select 5/Peripheral Chip Select Strobe

Master mode:

- Peripheral Chip Select 5 (O)—Used only when the peripheral-chip-select strobe is disabled (MCR[PCSSE]). Selects an SPI slave to receive data transmitted by the module.
- Peripheral Chip Select Strobe (O)—Used only when the peripheral-chip-select strobe is enabled (MCR[PCSSE]). Strobes an off-module peripheral-chip-select demultiplexer, which decodes the module's PCS signals other than PCS5, preventing glitches on the demultiplexer outputs.

Slave mode: Unused

42.3.6 SCK—Serial Clock

Master mode: Serial Clock (O)—Supplies a clock signal from the module to SPI slaves.

Slave mode: Serial Clock (I)—Supplies a clock signal to the module from an SPI master.

42.3.7 SIN—Serial Input

Master mode: Serial Input (I)—Receives serial data.

Slave mode: Serial Input (I)—Receives serial data.

42.3.8 SOUT—Serial Output

Master mode: Serial Output (O)—Transmits serial data.

Slave mode: Serial Output (O)—Transmits serial data.

42.4 Memory Map/Register Definition

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Any Write access to the POPR and RXFRn also results in a transfer error.

NOTE

While the module is in the running state, do not write to these registers:

- PISRn
- DIMRn, DPIRn, and SSRn
- CTAREn
- TRIG
- TSL and TS_CONF

DSPI memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (DSPI_MCR)	32	R/W	0000_4001h	42.4.1/1235
8	Transfer Count Register (DSPI_TCR)	32	R/W	0000_0000h	42.4.2/1238
C	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR0)	32	R/W	7800_0000h	42.4.3/1239
C	Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR0_SLAVE)	32	R/W	7800_0000h	42.4.4/1244
10	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR1)	32	R/W	7800_0000h	42.4.3/1239
10	Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR1_SLAVE)	32	R/W	7800_0000h	42.4.4/1244
14	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR2)	32	R/W	7800_0000h	42.4.3/1239
18	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR3)	32	R/W	7800_0000h	42.4.3/1239
1C	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR4)	32	R/W	7800_0000h	42.4.3/1239
20	Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR5)	32	R/W	7800_0000h	42.4.3/1239
2C	Status Register (DSPI_SR)	32	R/W	0201_0000h	42.4.5/1246
30	DMA/Interrupt Request Select and Enable Register (DSPI_RSER)	32	R/W	0000_0000h	42.4.6/1250
34	PUSH TX FIFO Register In Master Mode (DSPI_PUSHR)	32	R/W	0000_0000h	42.4.7/1253
34	PUSH TX FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)	32	R/W	0000_0000h	42.4.8/1255
38	POP RX FIFO Register (DSPI_POPR)	32	R	0000_0000h	42.4.9/1256
3C	Transmit FIFO Registers (DSPI_TXFR0)	32	R	0000_0000h	42.4.10/ 1256
40	Transmit FIFO Registers (DSPI_TXFR1)	32	R	0000_0000h	42.4.10/ 1256
44	Transmit FIFO Registers (DSPI_TXFR2)	32	R	0000_0000h	42.4.10/ 1256
48	Transmit FIFO Registers (DSPI_TXFR3)	32	R	0000_0000h	42.4.10/ 1256
7C	Receive FIFO Registers (DSPI_RXFR0)	32	R	0000_0000h	42.4.11/ 1257
80	Receive FIFO Registers (DSPI_RXFR1)	32	R	0000_0000h	42.4.11/ 1257

Table continues on the next page...

DSPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
84	Receive FIFO Registers (DSPI_RXFR2)	32	R	0000_0000h	42.4.11/ 1257
88	Receive FIFO Registers (DSPI_RXFR3)	32	R	0000_0000h	42.4.11/ 1257
BC	DSI Configuration Register 0 (DSPI_DSICR0)	32	R/W	0000_0000h	42.4.12/ 1258
C0	DSI Serialization Data Register 0 (DSPI_SDR0)	32	R	0000_0000h	42.4.13/ 1261
C4	DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)	32	R/W	0000_0000h	42.4.14/ 1261
C8	DSI Transmit Comparison Register 0 (DSPI_COMPR0)	32	R	0000_0000h	42.4.15/ 1262
CC	DSI Deserialization Data Register 0 (DSPI_DDR0)	32	R	0000_0000h	42.4.16/ 1262
D0	DSI Configuration Register 1 (DSPI_DSICR1)	32	R/W	0000_0000h	42.4.17/ 1263
D4	DSI Serialization Source Select Register 0 (DSPI_SSR0)	32	R/W	0000_0000h	42.4.18/ 1265
D8	DSI Parallel Input Select Register 0 (DSPI_PISR0)	32	R/W	0000_0000h	42.4.19/ 1265
DC	DSI Parallel Input Select Register 1 (DSPI_PISR1)	32	R/W	0000_0000h	42.4.20/ 1266
E0	DSI Parallel Input Select Register 2 (DSPI_PISR2)	32	R/W	0000_0000h	42.4.21/ 1267
E4	DSI Parallel Input Select Register 3 (DSPI_PISR3)	32	R/W	0000_0000h	42.4.22/ 1268
E8	DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)	32	R/W	0000_0000h	42.4.23/ 1269
EC	DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)	32	R/W	0000_0000h	42.4.24/ 1269
F0	DSI Serialization Data Register 1 (DSPI_SDR1)	32	R	0000_0000h	42.4.25/ 1270
F4	DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)	32	R/W	0000_0000h	42.4.26/ 1271
F8	DSI Transmit Comparison Register 1 (DSPI_COMPR1)	32	R	0000_0000h	42.4.27/ 1271
FC	DSI Deserialization Data Register 1 (DSPI_DDR1)	32	R	0000_0000h	42.4.28/ 1272
100	DSI Serialization Source Select Register 1 (DSPI_SSR1)	32	R/W	0000_0000h	42.4.29/ 1273
104	DSI Parallel Input Select Register 4 (DSPI_PISR4)	32	R/W	0000_0000h	42.4.30/ 1273
108	DSI Parallel Input Select Register 5 (DSPI_PISR5)	32	R/W	0000_0000h	42.4.31/ 1274

Table continues on the next page...

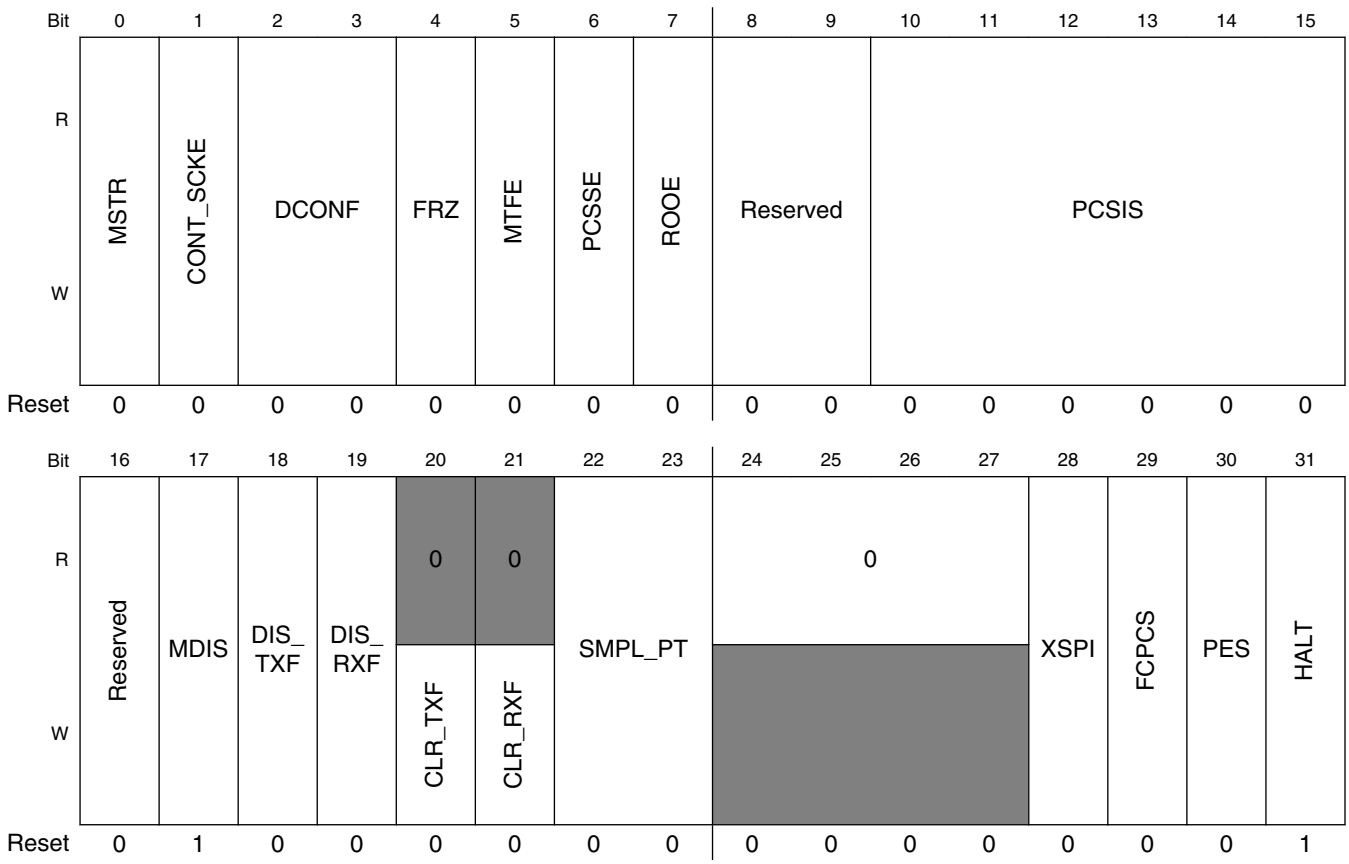
DSPI memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10C	DSI Parallel Input Select Register 6 (DSPI_PISR6)	32	R/W	0000_0000h	42.4.32/ 1275
110	DSI Parallel Input Select Register 7 (DSPI_PISR7)	32	R/W	0000_0000h	42.4.33/ 1276
114	DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)	32	R/W	0000_0000h	42.4.34/ 1277
118	DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)	32	R/W	0000_0000h	42.4.35/ 1278
11C	Clock and Transfer Attributes Register Extended (DSPI_CTARE0)	32	R/W	0000_0001h	42.4.36/ 1278
120	Clock and Transfer Attributes Register Extended (DSPI_CTARE1)	32	R/W	0000_0001h	42.4.36/ 1278
124	Clock and Transfer Attributes Register Extended (DSPI_CTARE2)	32	R/W	0000_0001h	42.4.36/ 1278
128	Clock and Transfer Attributes Register Extended (DSPI_CTARE3)	32	R/W	0000_0001h	42.4.36/ 1278
12C	Clock and Transfer Attributes Register Extended (DSPI_CTARE4)	32	R/W	0000_0001h	42.4.36/ 1278
130	Clock and Transfer Attributes Register Extended (DSPI_CTARE5)	32	R/W	0000_0001h	42.4.36/ 1278
13C	Status Register Extended (DSPI_SREX)	32	R	0000_0000h	42.4.37/ 1280
140	Trigger Register (DSPI_TRIG)	32	R/W	0000_0000h	42.4.38/ 1281
150	Time Slot Length Register (DSPI_TSL)	32	R/W	0000_0000h	42.4.39/ 1282
154	Time Slot Configuration Register (DSPI_TS_CONF)	32	R/W	0000_0000h	42.4.40/ 1283

42.4.1 Module Configuration Register (DSPI_MCR)

Contains bits to configure various attributes associated with the module operations. The HALT and MDIS bits can be changed at any time, but the effect takes place only on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed, while the module is in the Running state.

Address: 0h base + 0h offset = 0h



DSPI_MCR field descriptions

Field	Description
0 MSTR	Master/Slave Mode Select Enables either Master mode (if supported) or Slave mode (if supported) operation. 0 Enables Slave mode 1 Enables Master mode
1 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously.

Table continues on the next page...

DSPI_MCR field descriptions (continued)

Field	Description
	0 Continuous SCK disabled. 1 Continuous SCK enabled.
2–3 DCONF	Interface Configuration Selects among the different interface configurations of the module. 00 SPI 01 DSI 10 CSI 11 Reserved
4 FRZ	Freeze Enables transfers to be stopped on the next frame boundary when the device enters Debug mode. 0 Do not halt serial transfers in Debug mode. 1 Halt serial transfers in Debug mode.
5 MTFE	Modified Transfer Format Enable Enables a modified transfer format to be used. 0 Modified SPI transfer format disabled. 1 Modified SPI transfer format enabled.
6 PCSSE	Peripheral Chip Select Strobe Enable Enables the PCS5/ $\overline{\text{PCSS}}$ to operate as a PCS Strobe output signal. 0 PCS5/ $\overline{\text{PCSS}}$ is used as the Peripheral Chip Select[5] signal. 1 PCS5/ $\overline{\text{PCSS}}$ is used as an active-low PCS Strobe signal.
7 ROOE	Receive FIFO Overflow Overwrite Enable In the RX FIFO overflow condition, configures the module to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer, generating the overflow, is ignored or shifted into the shift register. 0 Incoming data is ignored. 1 Incoming data is shifted into the shift register.
8–9 Reserved	Always write the reset value to this field. This field is reserved.
10–15 PCSIS	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx. Refer to the chip-specific D SPI information for the number of PCS signals used in this chip. **NOTE:** The effect of this bit only takes place when module is enabled. Ensure that this bit is configured correctly before enabling the DSPI interface. 0 The inactive state of PCSx is low. 1 The inactive state of PCSx is high.
16 Reserved	This field is reserved.

Table continues on the next page...

DSPI_MCR field descriptions (continued)

Field	Description
17 MDIS	Module Disable Allows the clock to be stopped to the non-memory mapped logic in the module effectively putting it in a software-controlled power-saving state. The reset value of the MDIS bit is parameterized, with a default reset value of 1. When the module is used in Slave Mode, it is recommended to leave this bit 0, because a slave doesn't have control over master transactions. 0 Enables the module clocks. 1 Allows external logic to disable the module clocks.
18 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the module operates as a simplified double-buffered SPI. This bit can be written only when the MDIS bit is cleared. 0 TX FIFO is enabled. 1 TX FIFO is disabled.
19 DIS_RXF	Disable Receive FIFO When the RX FIFO is disabled, the receive part of the module operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0 RX FIFO is enabled. 1 RX FIFO is disabled.
20 CLR_TXF	Clear TX FIFO Flushes the TX FIFO. Writing a 1 to CLR_TXF clears the TX FIFO Counter. The CLR_TXF bit is always read as zero. 0 Do not clear the TX FIFO counter. 1 Clear the TX FIFO counter.
21 CLR_RXF	CLR_RXF Flushes the RX FIFO. Writing a 1 to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. **NOTE:** After every RX FIFO clear operation (MCR [CLR_RXF] = 0b1) following a RX FIFO overflow (SR [RFOF] = 0b1) scenario, immediately perform a single POP from the RX FIFO and discard the read data. The POP and discard operation should be completed before the reception of new incoming frame. 0 Do not clear the RX FIFO counter. 1 Clear the RX FIFO counter.
22–23 SMPL_PT	Sample Point Controls when the module master samples SIN in Modified Transfer Format. This field is valid only when CPHA bit in CTARn[CPHA] is 0. 00 0 protocol clock cycles between SCK edge and SIN sample 01 1 protocol clock cycle between SCK edge and SIN sample 10 2 protocol clock cycles between SCK edge and SIN sample 11 Reserved
24–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

DSPI_MCR field descriptions (continued)

Field	Description
28 XSPI	Extended SPI Mode This bit enables usage of CTARE (Command and Transfer Attribute Register Extended) Registers. CTARE registers allow the user to send up to 32 bit SPI frames. Command Cycling is also enabled which allows the user to send multiple Data Frames using a single Command Frame. When MCR[DIS_TXF] is asserted, the Extended SPI Mode cannot be used to transmit SPI frames which are more than 16 bits in size. 0 Normal SPI Mode. Frame size can be up to 16 bits. Command Cycling is not available in this mode. 1 Extended SPI Mode. Up to 32 bit SPI Frames along with Command Cycling is Enabled.
29 FCPCS	Fast Continuous PCS Mode. This bit enables the masking of “After SCK (t_{ASC})” and “PCS to SCK (t_{CSC})” delays when operating in Continuous PCS mode. This masking is not available if Continuous SCK mode is enabled. The individual delay masks are selected via bits MASC and MCSC of the PUSHR register. The firmware should select appropriate masks when providing continuous frames via the PUSHR register. 0 Normal or Slow Continuous PCS mode. Masking of delays is disabled. 1 Fast Continuous PCS mode. Delays masked via control bits in PUSHR register.
30 PES	Parity Error Stop Controls SPI operation when a parity error is detected in a received SPI frame. 0 SPI frame transmission continues. 1 SPI frame transmission stops.
31 HALT	Halt The HALT bit starts and stops frame transfers. See Start and Stop of Module transfers 0 Start transfers. 1 Stop transfers.

42.4.2 Transfer Count Register (DSPI_TCR)

TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. Do not write the TCR when the module is in the Running state.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SPI_TCNT																0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_TCR field descriptions

Field	Description
0–15 SPI_TCNT	SPI Transfer Counter Counts the number of SPI transfers the module makes. The SPI_TCNT field increments every time the last bit of an SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The Transfer Counter wraps around; incrementing the counter past 65535 resets the counter to zero.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

42.4.3 Clock and Transfer Attributes Register (In Master Mode) (DSPI_CTAR*n*)

CTAR registers are used to define different transfer attributes. Do not write to the CTAR registers while the module is in the Running state.

In Master mode, the CTAR registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays. In slave mode, a subset of the bitfields in CTAR0 and CTAR1 are used to set the slave transfer attributes.

When the module is configured as a SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the CTAR registers is used. When the module is configured as an SPI bus slave, it uses the CTAR0 register.

When the module is configured as a DSI master, the DSICTAS field in the DSI Configuration Register 0 (DSICR0), selects which of the CTAR register is used. When the module is configured as a DSI bus slave, the CTAR1 register is used.

In CSI Configuration, the transfer attributes are selected based on whether the current frame is SPI data or DSI data. SPI transfers in CSI Configuration follow the protocol described for SPI Configuration, and DSI transfers in CSI Configuration follow the protocol described for DSI Configuration.CSI Configuration is valid only in conjunction with master mode.

TSB mode sets some limitations on transfer attributes:

- Clock phase is forced to be CPHA = 1 and the CPHA bit setting has no effect.
- PCS lines are driven at the driving edge of the SCK clock together with SOUT, so PCS assertion and negation delays control is unavailable and PCSSCK, PASC, CSSCK and ASC fields have no effect.
- Delay after transfer can be set from 1 to 64 serial clocks with help of PDT and DT fields

Address: 0h base + Ch offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DBR	FMSZ				CPOL	CPHA	LSBFE	PCSSCK		PASC		PDT		PBR	
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CSSCK				ASC				DT				BR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_CTAR*n* field descriptions

Field	Description
0 DBR	Double Baud Rate Doubles the effective baud rate of the Serial Communications Clock (SCK). This field is used only in master mode. It effectively halves the Baud Rate division ratio, supporting faster frequencies, and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in the following table. See the BR field description for details on how to compute the baud rate.
1–4 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ value plus 1. Regardless of the transmission mode, the minimum valid frame size value is 4. There is a constraint on the minimum allowable Frame size, which depends on the ratio of the Register Read/Write clock to the Protocol clock. The minimum Frame size (FMSZ + 1) is determined by the following equation. Upper Ceiling must be applied for non-integer values.

Table 42-5. DSPI SCK Duty Cycle

DBR	CPHA	PBR	SCK Duty Cycle
0	any	any	50/50
1	0	00	50/50
1	0	01	33/66
1	0	10	40/60
1	0	11	43/57
1	1	00	50/50
1	1	01	66/33
1	1	10	60/40
1	1	11	57/43

0 The baud rate is computed normally with a 50/50 duty cycle.
1 The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.

Table continues on the next page...

DSPI_CTAR*n* field descriptions (continued)

Field	Description
	Minimum Frame Size = ((4 x f_p) + (3 x f_r)) / (n x f_r) f_p = Protocol clock frequency f_r = Register interface clock frequency n = (PBR x BR) / (1 + DBR) = Multiple of protocol clock required to get Baud (SCK) Clock However, the minimum frame size can never be less than 4. There is no constraint on the maximum programmable frame size. Also note that `fp' can be equal to, less than or greater than `fr' purely based on user requirement. When the DSPI operates in TSB mode, the FMSZ field value plus 1 is equal to the data frame bit number, where control of the PCS assertion switches from the DSICR0 to the DSICR1 register. If TSB Mode is operated when DSI is in 64-bit mode, then DSICR0[FMSZ4] also affects the switching of PCS assertion control.
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock polarities. When the Continuous Selection Format is selected, switching between clock polarities without stopping the module can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. **NOTE:** In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranteed. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode or TSB mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 LSBFE	LSB First Specifies whether the LSB or MSB of the frame is transferred first. When the DSPI operates in TSB configuration, this bit should be set to be compliant to the MSC specification. 0 Data is transferred MSB first. 1 Data is transferred LSB first.
8–9 PCSSCK	PCS to SCK Delay Prescaler Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description for information on how to compute the PCS to SCK Delay. Refer PCS to SCK Delay (t_{CSC}) for more details. In the TSB mode, the PCSSCK field has no effect. 00 PCS to SCK Prescaler value is 1. 01 PCS to SCK Prescaler value is 3. 10 PCS to SCK Prescaler value is 5. 11 PCS to SCK Prescaler value is 7.

Table continues on the next page...

DSPI_CTAR*n* field descriptions (continued)

Field	Description
10–11 PASC	After SCK Delay Prescaler Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. See the ASC field description for information on how to compute the After SCK Delay. In the TSB mode, the PASC field has no effect. Refer After SCK Delay (t_{ASC}) for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
12–13 PDT	Delay after Transfer Prescaler Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. In the TSB mode the PDT field defines two most-signficant bits of the Delay after Transfer. See the DT field description for details on how to compute the Delay after Transfer. Refer Delay after Transfer (t_{DT}) for more details. 00 Delay after Transfer Prescaler value is 1. 01 Delay after Transfer Prescaler value is 3. 10 Delay after Transfer Prescaler value is 5. 11 Delay after Transfer Prescaler value is 7.
14–15 PBR	Baud Rate Prescaler Selects the prescaler value for the baud rate. This field is used only in master mode. The baud rate is the frequency of the SCK. The protocol clock is divided by the prescaler value before the baud rate selection takes place. See the BR field description for details on how to compute the baud rate. 00 Baud Rate Prescaler value is 2. 01 Baud Rate Prescaler value is 3. 10 Baud Rate Prescaler value is 5. 11 Baud Rate Prescaler value is 7.
16–19 CSSCK	PCS to SCK Delay Scaler Selects the scaler value for the PCS to SCK delay. This field is used only in master mode. In the TSB mode the field has no effect. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{CSC} = (1/f_P) \times PCSSCK \times CSSCK$. The following table lists the delay scaler values.

Table 42-6. Delay Scaler Encoding

Field Value	Delay Scaler Value
0000	2
0001	4
0010	8
0011	16
0100	32

Table continues on the next page...

DSPI_CTAR*n* field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
Table 42-6. Delay Scaler Encoding (continued)
<table>
<tr><th>Field Value</th><th>Delay Scaler Value</th></tr>
<tr><td>0101</td><td>64</td></tr>
<tr><td>0110</td><td>128</td></tr>
<tr><td>0111</td><td>256</td></tr>
<tr><td>1000</td><td>512</td></tr>
<tr><td>1001</td><td>1024</td></tr>
<tr><td>1010</td><td>2048</td></tr>
<tr><td>1011</td><td>4096</td></tr>
<tr><td>1100</td><td>8192</td></tr>
<tr><td>1101</td><td>16384</td></tr>
<tr><td>1110</td><td>32768</td></tr>
<tr><td>1111</td><td>65536</td></tr>
</table>
Refer PCS to SCK Delay (t_{CSC}) for more details.
</td></tr>
<tr><td>20–23
ASC</td><td>After SCK Delay Scaler

Selects the scaler value for the After SCK Delay. This field is used only in master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the protocol clock period, and it is computed according to the following equation:

$t_{ASC} = (1/f_P) \times PASC \times ASC$

See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer After SCK Delay (t_{ASC}) for more details.</td></tr>
<tr><td>24–27
DT</td><td>Delay After Transfer Scaler

Selects the Delay after Transfer Scaler. This field is used only in master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame.

In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period, The Delay after Transfer is a multiple of the protocol clock period, and it is computed according to the following equation:

$t_{DT} = (1/f_P) \times PDT \times DT$

See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values.

In the TSB mode, the Delay after Transfer is equal to a number formed by concatenation of the PDT and DT fields plus 1 of the SCK clock periods. Refer Delay after Transfer (t_{DT}) for more details.</td></tr>
<tr><td>28–31
BR</td><td>Baud Rate Scaler

Selects the scaler value for the baud rate. This field is used only in master mode. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is computed according to the following equation:

SCK baud rate = (f_P /PBR) x [(1+DBR)/BR]

The following table lists the baud rate scaler values.</td></tr>
</table>

Table continues on the next page...

DSPI_CTAR*n* field descriptions (continued)

Field	Description

Table 42-7. Baud Rate Scaler

CTARn[BR]	Baud Rate Scaler Value
0000	2
0001	4
0010	6
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

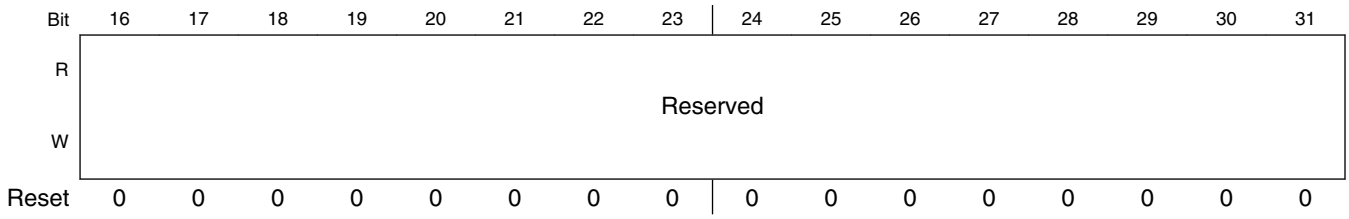
42.4.4 Clock and Transfer Attributes Register (In Slave Mode) (DSPI_CTAR*n*_SLAVE)

When the module is configured as an SPI bus slave, the CTAR0 register is used.

When the module is configured as a DSI master, the DSICTAS field in the DSI Configuration Register 0 (DSICR0), selects which of the CTAR register is used. When the module is configured as a DSI bus slave, the CTAR1 register is used.

Address: 0h base + Ch offset + (4d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	FMSZ					CPOL	CPHA	PE	PP	FMSZ5	Reserved					
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0



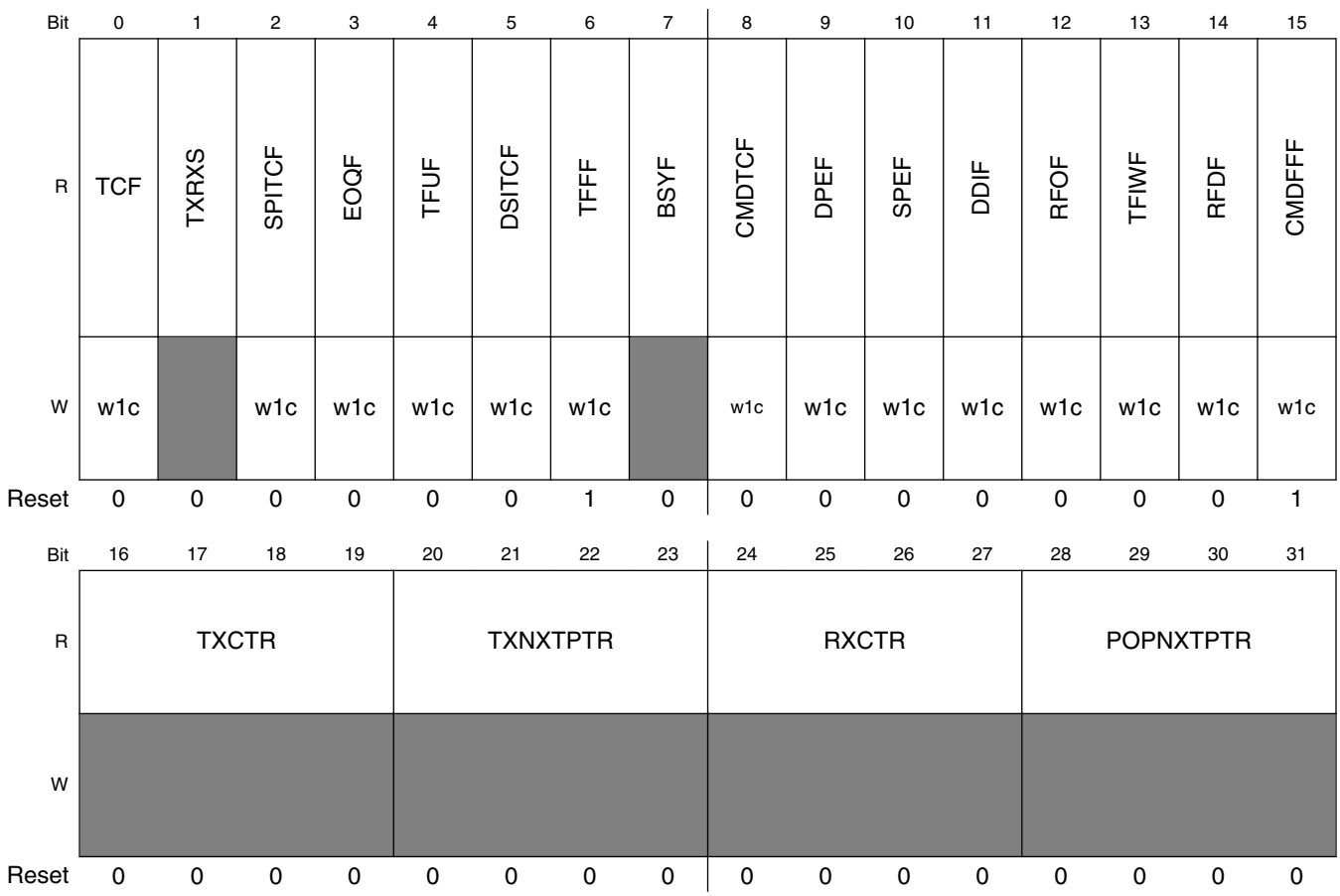
DSPI_CTAR*n*_SLAVE field descriptions

Field	Description
0–4 FMSZ	Frame Size The number of bits transferred per frame is equal to the concatenated field {CTARn_SLAVE[FMSZ5], CTARn_SLAVE[FMSZ]} value plus 1. Note that the minimum valid value of frame size is 4.
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). **NOTE:** In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranteed. 0 The inactive state value of SCK is low. 1 The inactive state value of SCK is high.
6 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode or TSB mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0 Data is captured on the leading edge of SCK and changed on the following edge. 1 Data is changed on the leading edge of SCK and captured on the following edge.
7 PE	Parity Enable Enables parity bit transmission and reception for the frame. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of last data bit in frame, parity checked for received frame.
8 PP	Parity Polarity Controls polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame.
9 FMSZ5	MSB of Frame Size when DSI is used in 64-bit Mode The number of bits transferred per frame is equal to the concatenated field {CTARn_SLAVE[FMSZ[5]], CTARn_SLAVE[FMSZ]} value plus 1.
10–31 Reserved	This field is reserved.

42.4.5 Status Register (DSPI_SR)

SR contains status and flag bits. The bits reflect the status of the module and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the SR by writing a 1 to them. Writing a 0 to a flag bit has no effect. This register may not be writable in Module Disable mode due to the use of power saving mechanisms.

Address: 0h base + 2Ch offset = 2Ch



DSPI_SR field descriptions

Field	Description
0 TCF	Transfer Complete Flag Indicates that all bits in a frame have been shifted out. TCF remains set until it is cleared by writing a 1 to it. 0 Transfer not complete. 1 Transfer complete.

Table continues on the next page...

DSPI_SR field descriptions (continued)

Field	Description
1 TXRXS	TX and RX Status Reflects the run status of the module. 0 Transmit and receive operations are disabled (The module is in Stopped state). 1 Transmit and receive operations are enabled (The module is in Running state).
2 SPITCF	SPI Frame Transfer Complete Flag. This bit functions similar to TCF except that this bit is asserted only on SPI frame transmission completion in the following modes: • CSI • TSB in CSI • ITSB This flag will not be asserted when the module is used in SPI or DSI Modes. 0 SPI frame transfer is not complete. 1 SPI frame transfer is complete.
3 EOQF	End of Queue Flag Indicates that the last entry in a queue has been transmitted when the module is in Master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. The EOQF bit remains set until cleared by writing a 1 to it. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0 EOQ is not set in the executing command. 1 EOQ is set in the executing SPI command.
4 TFUF	Transmit FIFO Underflow Flag Indicates an underflow condition in the TX FIFO. The transmit underflow condition is detected only for DSPI blocks operating in Slave mode and SPI configuration. TFUF is set when the TX FIFO of the module operating in SPI Slave mode is empty and an external SPI master initiates a transfer. The TFUF bit remains set until cleared by writing 1 to it. 0 No TX FIFO underflow. 1 TX FIFO underflow has occurred.
5 DSITCF	DSI Frame Transfer Complete Flag. This bit functions similar to TCF except that this bit is asserted only on DSI frame transmission completion in the following modes: • CSI • TSB in CSI • ITSB This flag will not be asserted when the module is in SPI or DSI mode. 0 DSI frame transfer is not complete. 1 DSI frame transfer is complete.
6 TFFF	Transmit FIFO Fill Flag Indicates whether there is an available location to be filled in the FIFO. Either a DMA request or an interrupt indication can be used to add another entry to the FIFO. Note that this bit is set if at least one location is free in the FIFO. The TFFF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller to the TX FIFO full request.

Table continues on the next page...

DSPI_SR field descriptions (continued)

Field	Description
	NOTE: The reset value of this bit is 0 when the module is disabled,(MCR[MDIS]=1). 0 TX FIFO is full. 1 TX FIFO is not full.
7 BSYF	Busy Flag. This bit is valid only when DSPI_MCR[XSPI] is enabled. Indicates that the current Command Frame is being used for transmiting multiple data frames. This bit is not set for the last Data Frame of a Cyclic command Transfer or when DSPI_CTARE[DTCP] = 1. Refer Command First In First Out (CMD FIFO) Buffering Mechanism for more details. 0 No Cyclic Command Transfer in Progress. 1 Cyclic Command Transfer is in progress. Current Data Frame is not the last data frame for on-going cyclic command transfer..
8 CMDTCF	Command Transfer Complete Flag. Indicates that the last Data frame for the current Cyclic Command has been transmitted. Hence this bit is set only for the last Data Frame of a Cyclic Command Transfer or when DSPI_CTARE[DTCP] = 1. The bit remains set until it is cleared by writing a ‘1’ to it. 0 Data Transfer by current Command not complete. 1 Data Transfer by current Command is complete.
9 DPEF	DSI Parity Error Flag Indicates that a DSI frame with parity error had been received. The bit remains set until it is cleared by writing a 1 to it. 0 No parity error. 1 Parity error has occurred.
10 SPEF	SPI Parity Error Flag Indicates that a SPI frame with parity error had been received. The bit remains set until it is cleared by writing a 1 to it. 0 No parity error. 1 Parity error has occurred.
11 DDIF	DSI Data Received with Active Bits Indicates that a DSI frame had been received with bits selected by DIMR with active polarity, defined by the DPIR register. The bit remains set until it is cleared by writing a 1 to it. 0 No DSI data with active bits was received. 1 DSI data with active bits was received.
12 RFOF	Receive FIFO Overflow Flag Indicates an overflow condition in the RX FIFO. The field is set when the RX FIFO and shift register are full and a transfer is initiated. The bit remains set until it is cleared by writing a 1 to it. 0 No Rx FIFO overflow. 1 Rx FIFO overflow has occurred.
13 TFIWF	Tranmit FIFO Invalid Write Flag

Table continues on the next page...

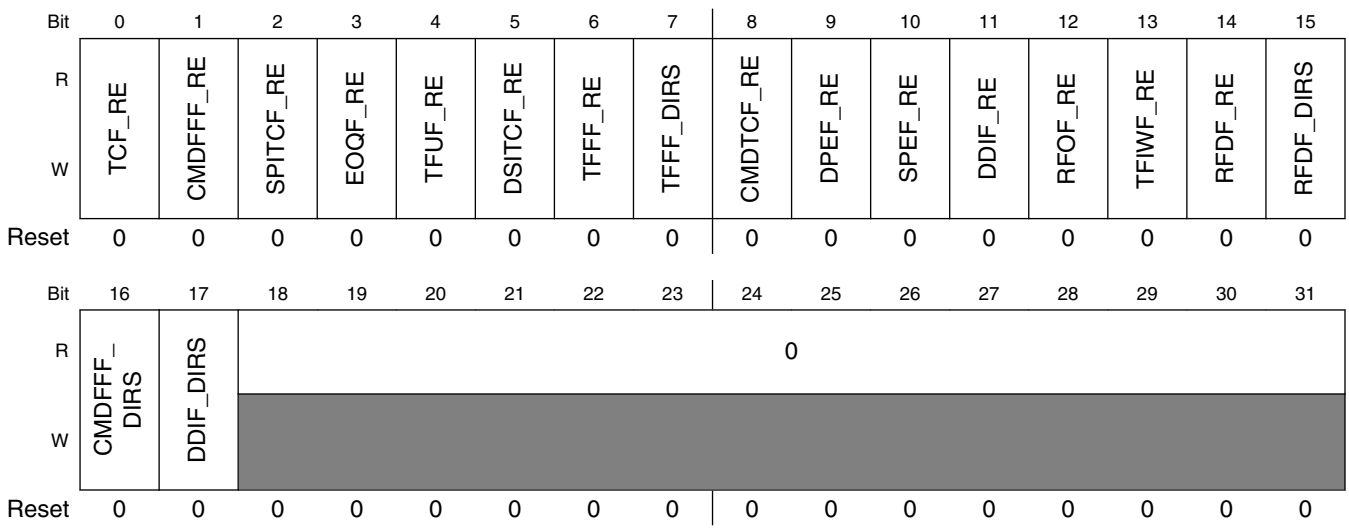
DSPI_SR field descriptions (continued)

Field	Description
	Indicates Data Write on TX FIFO while CMD FIFO is empty. Without a Command, the Data entries present in TXFIFO are invalid. This bit remains set until it is cleared by writing a '1' to it. 0 No Invalid Data present in TX FIFO. 1 Invalid Data present in TX FIFO since CMD FIFO is empty.
14 RFDF	Receive FIFO Drain Flag Indicates whether there is an available location to be drained from the FIFO. Either a DMA request or an interrupt indication can be used to read from the FIFO. Note that this bit is set if at least one location can be read from the FIFO. The RFDF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller when the RX FIFO is empty. 0 RX FIFO is empty. 1 RX FIFO is not empty.
15 CMDFFF	Command FIFO Fill Flag Indicates whether there is an available location to be filled in the FIFO. Either a DMA request or an interrupt indication can be used to add another entry to the FIFO. Note that this bit is set if at least one location is free in the FIFO. The CMDFFF is cleared by writing a '1' to it or by acknowledgement from the DMA controller to the CMD FIFO full request. 0 CMD FIFO is full. 1 CMD FIFO is not full.
16–19 TXCTR	TX FIFO Counter Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the PUSHR is written. The TXCTR is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
20–23 TXNXTPTR	Transmit Next Pointer Indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register.
24–27 RXCTR	RX FIFO Counter Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
28–31 POPNXTPTR	Pop Next Pointer Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read.

42.4.6 DMA/Interrupt Request Select and Enable Register (DSPI_RSER)

RSER controls DMA and interrupt requests. Do not write to the RSER while the module is in the Running state.

Address: 0h base + 30h offset = 30h



DSPI_RSER field descriptions

Field	Description
0 TCF_RE	Transmission Complete Request Enable Enables TCF flag in the SR to generate an interrupt request. 0 TCF interrupt requests are disabled. 1 TCF interrupt requests are enabled.
1 CMDFFF_RE	Command FIFO Fill Flag Request Enable. Enables the CMDFFF flag in the SR to generate a request. The CMDFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 CMDFFF interrupts or DMA requests are disabled. 1 CMDFFF interrupts or DMA requests are enabled.
2 SPITCF_RE	SPI Frame Transmission Complete Request Enable. The SPITCF_RE bit enables SPITCF flag in the SR to generate an interrupt request. 0 SPITCF interrupt requests are disabled. 1 SPITCF interrupt requests are enabled.
3 EOQF_RE	Finished Request Enable

Table continues on the next page...

DSPI_RSER field descriptions (continued)

Field	Description
	Enables the EOQF flag in the SR to generate an interrupt request. 0 EOQF interrupt requests are disabled. 1 EOQF interrupt requests are enabled.
4 TFUF_RE	Transmit FIFO Underflow Request Enable Enables the TFUF flag in the SR to generate an interrupt request. 0 TFUF interrupt requests are disabled. 1 TFUF interrupt requests are enabled.
5 DSITCF_RE	DSI Frame Transmission Complete Request Enable. The DSITCF_RE bit enables DSITCF flag in the SR to generate an interrupt request. 0 DSITCF interrupt requests are disabled. 1 DSITCF interrupt requests are enabled.
6 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0 TFFF interrupts or DMA requests are disabled. 1 TFFF interrupts or DMA requests are enabled.
7 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When SR[TFFF] and RSER[TFFF_RE] are set, this field selects between generating an interrupt request or a DMA request. 0 TFFF flag generates interrupt requests. 1 TFFF flag generates DMA requests.
8 CMDTCF_RE	Command Transmission Complete Request Enable. The CMDTCF_RE bit enables CMDTCF flag in the SR to generate an interrupt request. 0 CMDTCF interrupt requests are disabled. 1 CMDTCF interrupt requests are enabled.
9 DPEF_RE	DSI Parity Error Request Enable Enables the DPEF flag in the SR to generate an interrupt request. 0 DPEF interrupt requests are disabled. 1 DPEF interrupt requests are enabled.
10 SPEF_RE	SPI Parity Error Request Enable Enables the SPEF flag in the SR to generate an interrupt request. 0 SPEF interrupt requests are disabled. 1 SPEF interrupt requests are enabled.
11 DDIF_RE	DSI data received with active bits Request Enable Enables the SR[DDIF] flag to generate an interrupt or DMA request.

Table continues on the next page...

DSPI_RSER field descriptions (continued)

Field	Description
	0 Disabled 1 Enabled
12 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the SR to generate an interrupt request. 0 RFOF interrupt requests are disabled. 1 RFOF interrupt requests are enabled.
13 TFIWF_RE	Transmit FIFO Invalid Write Request Enable. Enables the TFIWF flag in the SR to generate an interrupt request. 0 TFIWF interrupt requests are disabled. 1 TFIWF interrupt requests are enabled.
14 RFDF_RE	Receive FIFO Drain Request Enable Enables the RFDF flag in the SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 RFDF interrupt or DMA requests are disabled. 1 RFDF interrupt or DMA requests are enabled.
15 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the SR is set, and the RFDF_RE bit in the RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request. 1 DMA request.
16 CMDFFF_DIRS	Command FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the CMDFFF flag bit in the SR is set, and the CMDFFF_RE bit in the RSER is set, the CMDFFF_DIRS bit selects between generating an interrupt indication or a DMA request. 0 CMDFFF flag generates interrupt requests. 1 CMDFFF flag generates DMA requests.
17 DDIF_DIRS	DSI data received with active bits - DMA or Interrupt Request Select. Selects between generating a DMA request or an interrupt request. When the DDIF flag bit in the SR is set, and the DDIF_RE bit in the RSER is set, the DDIF_DIRS bit selects between generating an interrupt request or a DMA request. 0 Interrupt request. 1 DMA request.
18–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

42.4.7 PUSH TX FIFO Register In Master Mode (DSPI_PUSHR)

Specifies data to be transferred to the TX FIFO and CMD FIFO. User must write 16-bits data into TXDATA field. An 8- or 16-bit write access to the TXDATA field transfers 16 bits of data bus to the TX FIFO. A write access to the command fields transfers the 16 bits of command information to the CMD FIFO. In Master mode, the register transfers 16 bits of data to the TX FIFO and 16 bits of command information to the CMD FIFO.

If Extended SPI Mode is disabled (MCR[XSPI]), the TX FIFO and CMD FIFO must be filled simultaneously. In other words, you must perform write accesses to both the data and command fields for every PUSHR operation. With Extended SPI Mode disabled and both the TX FIFO and CMD FIFO are written to and read from simultaneously, they behave as a single 32 bit FIFO. When Extended SPI mode is enabled (MCR[XSPI]), the TX FIFO and CMD FIFO can be written to independently.

A read access of PUSHR returns the topmost TX FIFO and CMD FIFO entries concatenated.

When the module is disabled, writing to this register does not update the FIFO. Therefore, any reads performed while the module is disabled return the last PUSHR write performed while the module was still enabled.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CONT	CTAS			EOQ	CTCNT	PE_MASC	PP_MCSC	Reserved		PCS					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_PUSHR field descriptions

Field	Description
0 CONT	Continuous Peripheral Chip Select Enable Selects a continuous selection format. The bit is used in SPI Master mode. The bit enables the selected PCS signals to remain asserted between transfers.

Table continues on the next page...

DSPI_PUSHR field descriptions (continued)

Field	Description
	0 Return PCSn signals to their inactive state between transfers. 1 Keep PCSn signals asserted between transfers.
1–3 CTAS	Clock and Transfer Attributes Select Selects which CTAR to use in master mode to specify the transfer attributes for the associated SPI frame. In SPI Slave mode, CTAR0 is used. See the chip specific section for details to determine how many CTARs this device has. You should not program a value in this field for a register that is not present. 000 CTAR0 001 CTAR1 010 CTAR2 011 CTAR3 100 CTAR4 101 CTAR5 110 Reserved 111 Reserved
4 EOQ	End Of Queue Host software uses this bit to signal to the module that the current SPI transfer is the last in a queue. At the end of the transfer, the EOQF bit in the SR is set. 0 The SPI data is not the last data to transfer. 1 The SPI data is the last data to transfer.
5 CTCNT	Clear Transfer Counter Clears the TCNT field in the TCR register. The TCNT field is cleared before the module starts transmitting the current SPI frame. 0 Do not clear the TCR[TCNT] field. 1 Clear the TCR[TCNT] field.
6 PE_MASC	Parity Enable or Mask T_{ASC} delay in the current frame PE – This bit enables parity bit transmission and parity reception check for the SPI frame. MASC - The current frame has the “after SCK” delay masked if this bit is asserted. See Fast Continuous Selection Format for more details. **NOTE:** This bit is used as Mask T_{ASC} in the Fast Continuous PCS mode when MCR[FCPCS] is set. 0 PE - No parity bit included/checked. MASC - T_{ASC} delay is not masked and the current frame has the after SCK delay. 1 PE - Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame. MASC - T_{ASC} delay is masked in the current frame.
7 PP_MCSC	Parity Polarity or Mask T_{CSC} delay in the next frame PP - It controls the polarity of the parity bit transmitted and checked. MCSC - The next frame has the “PCS to SCK” delay masked if this bit is asserted. See Fast Continuous Selection Format for more details. **NOTE:** This bit is used as Mask T_{CSC} in the Fast Continuous PCS mode when MCR[FCPCS] is set.

Table continues on the next page...

DSPI_PUSHR field descriptions (continued)

Field	Description
	0 PP - Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. MCSC - T_{CSC} delay is not masked and the next frame has the PCS to SCK delay. 1 PP - Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame. MCSC - T_{CSC} delay is masked in the next frame.
8–9 Reserved	Always write the reset value to this field. This field is reserved.
10–15 PCS	Select which PCS signals are to be asserted for the transfer. Refer to the chip-specific DSPI information for the number of PCS signals used in this chip. 0 Negate the PCS[x] signal. 1 Assert the PCS[x] signal.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

42.4.8 PUSH TX FIFO Register In Slave Mode (DSPI_PUSHR_SLAVE)

Specifies data to be transferred to the TX FIFO in slave mode. An 8- or 16-bit write access to PUSHR transfers the 16-bit TXDATA field to the TX FIFO.

When Extended SPI mode is enabled (MCR[XSPI]), up to 32-bit SPI frames may be queued for transmission or reception.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXDATA															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

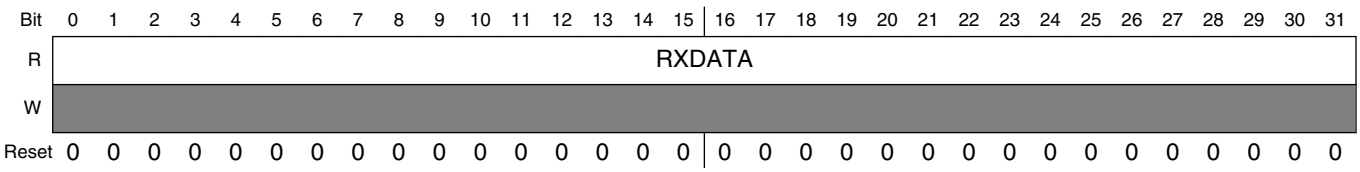
DSPI_PUSHR_SLAVE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

42.4.9 POP RX FIFO Register (DSPI_POPR)

POPR is used to read the RX FIFO. Eight- or sixteen-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register will generate a Transfer Error.

Address: 0h base + 38h offset = 38h



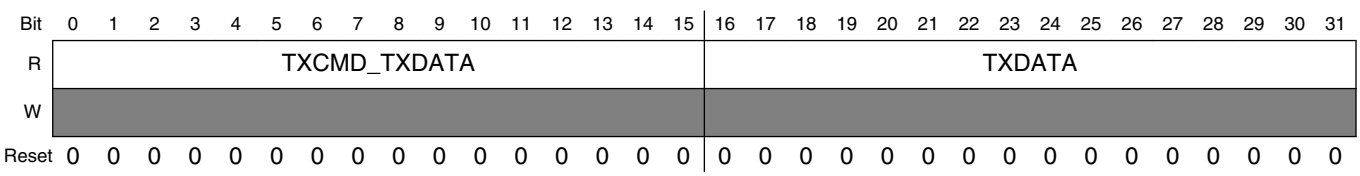
DSPI_POPR field descriptions

Field	Description
0–31 RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

42.4.10 Transmit FIFO Registers (DSPI_TXFR*n*)

TXFRn registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the TXFRx registers does not alter the state of the TX FIFO. When the module is operating in Extended SPI mode, reading TXFRn registers is invalid.

Address: 0h base + 3Ch offset + (4d × i), where i=0d to 3d



DSPI_TXFR*n* field descriptions

Field	Description
0–15 TXCMD_ TXDATA	Transmit Command or Transmit Data In Master mode the TXCMD field contains the command that sets the transfer attributes for the SPI data. In Slave mode, this field is reserved.

Table continues on the next page...

DSPI_TXFR*n* field descriptions (continued)

Field	Description
16–31 TXDATA	Transmit Data Contains the SPI data to be shifted out.

42.4.11 Receive FIFO Registers (DSPI_RXFR*n*)

RXFRn provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The RXFR registers are read-only. Reading the RXFRx registers does not alter the state of the RX FIFO. The field MCR[MDIS] must be 0 when RXFR is read.

Address: 0h base + 7Ch offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RXDATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_RXFR*n* field descriptions

Field	Description
0–31 RXDATA	Receive Data Contains the received SPI data.

42.4.12 DSI Configuration Register 0 (DSPI_DSICR0)

Selects various attributes associated with the following configurations:

- DSI
- TSB
- ITSB

Do not write to the DSICR0 while the module is in the Running state.

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MTOE	FMSZ4	MTOCNT						FMSZ5	0	ITSB	TSBC	TXSS	TPOL	TRRE	CID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DCONT	DSICTAS			DMS	PES	PE	PP	DPCSx							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DSICR0 field descriptions

Field	Description
0 MTOE	Multiple Transfer Operation Enable Enables multiple DSPIs to be connected in a parallel or serial configuration. The MTOE and TSB bits should not be set simulteneously. 0 Multiple transfer operation disabled. 1 Multiple transfer operation enabled.
1 FMSZ4	MSB of the frame size in master mode. If the bit is set, 16 is added to the frame size, as defined by the CTARn[FMSZ] field. The CTARx register is selected by the DSICTAS field.
2–7 MTOCNT	Multiple Transfer Operation Count Selects the number of bits to be shifted out during a transfer in Multiple Transfer Operation. The field sets the number of SCK cycles that the bus master must generate to complete the transfer. The number of SCK cycles used is one more than the value in the MTOCNT field. The number of SCK cycles defined by MTOCNT must be equal to or greater than the frame size. When TSBC is set, MTOCNT has no effect.
8 FMSZ5	MSB of the frame size in master mode when DSI is used in 64-bit mode.

Table continues on the next page...

DSPI_DSICR0 field descriptions (continued)

Field	Description
	If the bit is set, 32 is added to the frame size, as defined by the concatenation of {DSICR0[FMSZ[4]] , CTARn[FMSZ]} fields. The CTARn register is selected by the DSICTAS field. Hence the final concatenated frame size using this bit would be {DSICR0[FMSZ[5]], DSICR0[FMSZ[4]] , CTARn[FMSZ]}. This field is only valid when DSICR1[DSI64E] is enabled.
9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 ITSB	Interleaved TSB mode. This bit enables the Interleaved TSB mode of operation. When enabled, there is no priority among frames from the SPI/DSI. DSI frames are always sent when either the previous transmission was a SPI frame or if the TX_FIFO is empty. Frames are transmitted on every trigger whose source is selected by the TRG bit setting. The ITSB mode requires the DSICR0[TSBC] bit to be set and should be written only when MCR[HALT] bit is asserted. If ITSB bit is set without setting DSICR0[TSBC], then the module operates in the Normal Mode depending on MCR[DCONF] bit. See Interleaved TSB (ITSB) Mode for more details. 0 Interleaved mode is disabled. 1 Interleaved mode is enabled.
11 TSBC	Timed Serial Bus Configuration. Enables the Timed Serial Bus interface configuration and allows t_{DT} to be programmable. 0 Timed Serial Bus Configuration disabled. 1 Timed Serial Bus Configuration enabled.
12 TXSS	Transmit Data Source Select. Selects the source of data to be serialized. The source can be either data from host software written to the DSI Alternate Serialization Data Register (ASDR0/1) or Parallel Input pin states latched into the DSI Serialization Data Register (SDR0/1). 0 Source of serialized data is the SDR. 1 Source of serialized data is the ASDR.
13 TPOL	Trigger Polarity Selects the active edge of the hardware trigger input signal (HT), initiating DSI frames transfer. 0 Falling edge initiates a transfer. 1 Rising edge initiates a transfer.
14 TRRE	Trigger Reception Enable Enables the module to initiate a DSI frames transfer with an external trigger signal. **NOTE:** When TRIG[ENABLE] is set, the TRRE bit is given by TRIG[TRRE] which overrides DSICR0[TRRE]. 0 Trigger signal reception disabled. 1 Trigger signal reception enabled.
15 CID	Change In Data Transfer Enable Enables a change in serialization data to initiate a DSI frames transfer in DSI and CSI configurations. When the CID bit is set, DSI frames are initiated when the current DSI data differs from the previous DSI data shifted out.

Table continues on the next page...

DSPI_DSICR0 field descriptions (continued)

Field	Description
	NOTE: When TRIG[ENABLE] is set, the CID bit is given by TRIG[CID] which overrides DSICR0[CID].
16 DCONT	DSI Continuous Peripheral Chip Select Enable Enables the PCS signals to remain asserted between transfers. The DCONT bit affects the PCS signals only in DSI master mode. When the TSBC bit is set, DCONT has no effect. 0 Return Peripheral Chip Select signals to their inactive state after the transfer is complete. 1 Keep Peripheral Chip Select signals asserted after transfer is complete.
17–19 DSICTAS	DSI Clock and Transfer Attributes Select Selects the CTAR register to use to provide transfer attributes for DSI frames. The DSICTAS field is used in DSI master mode. In DSI slave mode, CTAR1 is always selected.
20 DMS	Data Match Stop If set, stops DSI frame transmissions if DDIF flag is set in the SR register. 0 DDIF flag does not have an effect on DSI frame transmissions. 1 DDIF flag stops DSI frame transmissions.
21 PES	Parity Error Stop If set, stops DSI operation if the parity error happened in the received DSI frame. 0 Parity error does not stop DSI frame transmissions. 1 Parity error stops all DSI frame transmissions.
22 PE	Parity Enable Enables parity bit transmission and parity reception check for the DSI frames. 0 No parity bit included/checked. 1 Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame.
23 PP	Parity Polarity Controls the polarity of the parity bit transmitted and checked. 0 Even Parity: the number of 1 bits in the transmitted frame is even. The SR[DPEF] bit is set if in the received frame number of 1 bits is odd. 1 Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[DPEF] bit is set if in the received frame number of 1 bits is even.
24–31 DPCSx	DSI Peripheral Chip Select 0–7 Selects which of the PCS signals to assert during a DSI master mode transfer. 0 Negate PCS[x]. 1 Assert PCS[x].

42.4.13 DSI Serialization Data Register 0 (DSPI_SDR0)

SDR0 contains the states of the 32 LSB Parallel Input signals. The states of the Parallel Input signals are latched into the SDR0 on the rising edge of every bus clock. The SDR0 is read-only. When the TXSS bit in the DSICR0 is cleared, the data in the SDR0 is used as the source of the DSI frames. When DSI 64-bit mode is enabled (DSICR1[DSI64E]), the concatenation of {SDR1, SDR0} provides the 64-bit Serialization Data value.

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SER_DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_SDR0 field descriptions

Field	Description
0–31 SER_DATA	Serialized Data Contains the signal states of the 32 LSB Parallel Input signals.

42.4.14 DSI Alternate Serialization Data Register 0 (DSPI_ASDR0)

ASDR0 is used by host software to write the 32 LSB of the data to be serialized. When the TXSS bit in the DSICR0 is set, the data in the ASDR0 is the source of the DSI frames. Writes to the ASDR0 take effect on the next frame boundary. When DSI 64-bit mode is enabled (DSICR1[DSI64E]), the concatenation of {ASDR1, ASDR0} provides the 64-bit Alternate Serialization Data value.

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ASER_DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_ASDR0 field descriptions

Field	Description
0–31 ASER_DATA	Alternate Serialized Data Holds the alternate 32 LSB of the data to be serialized.

42.4.15 DSI Transmit Comparison Register 0 (DSPI_COMPR0)

COMPR0 holds a copy of the 32 LSB of the last transmitted DSI data. COMPR0 is read-only. DSI data is transferred to this register as it is loaded into the TX Shift Register. When DSI 64-bit mode is enabled (DSICR1[DSI64E]), the concatenation of {COMPR1, COMPR0} provides the 64-bit Transmit Comparison value.

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_COMPR0 field descriptions

Field	Description
0–31 COMP_DATA	Compare Data Holds the 32 LSB of the last serialized DSI data.

42.4.16 DSI Deserialization Data Register 0 (DSPI_DDR0)

DDR0 holds the 32 LSB signal states for the Parallel Output signals. DDR0 is read-only and host software can read data from incoming DSI frames. When DSI 64-bit mode is enabled (DSICR1[DSI64E]), the concatenation of {DDR1, DDR0} provides the 64-bit Deserialization Data value.

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DESER_DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DDR0 field descriptions

Field	Description
0–31 DESER_DATA	Deserialized Data Holds 32 LSB of deserialized data that is presented as signal states to the Parallel Output signals.

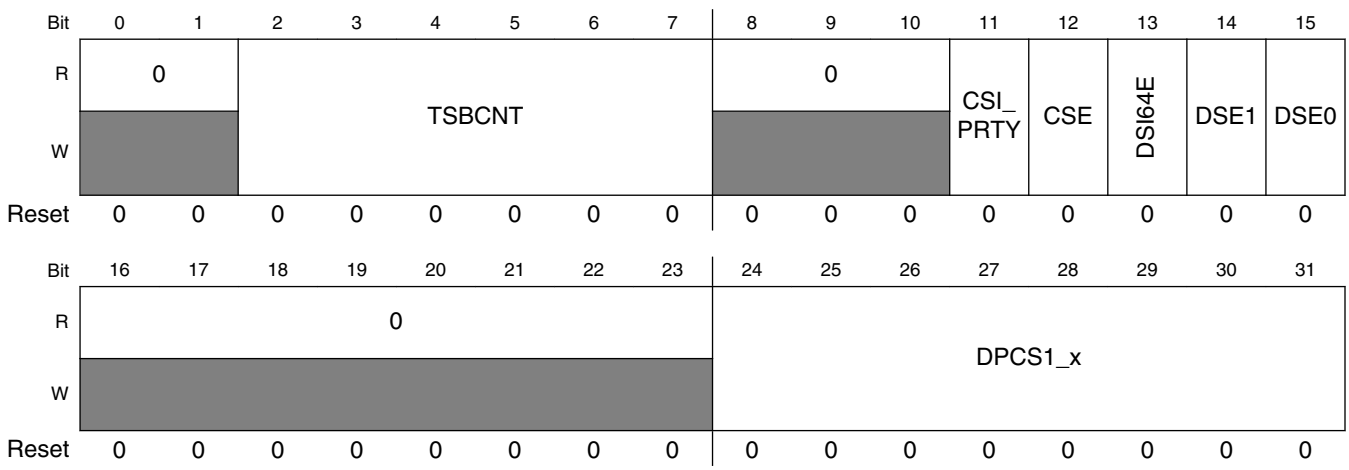
42.4.17 DSI Configuration Register 1 (DSPI_DSICR1)

Selects various attributes associated with the following configurations:

- DSI
- TSB
- ITSB

The user must not write to the DSICR1 while the module is in the Running state.

Address: 0h base + D0h offset = D0h



DSPI_DSICR1 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 TSBCNT	Timed Serial Bus Operation Count When TSB is enabled (DSICR0[TSBC]), TSBCNT defines the length of the data frame. When DSI 64-bit mode is enabled (DSI64E), the length value can vary from 3 to 63 (000011b to 111111b); otherwise, it can vary from 3 to 31 (000011b to 011111b). The length value specifies the number of data bits to be shifted out during a transfer in TSB mode. The number of data bits in the data frame is one more than the value in the TSBCNT field.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 CSI_PRTY	CSI Priority This field allows the user to configure the priority of SPI and DSI frames when module is operating in CSI Mode. This is not applicable when DSICR0[ITSB] is enabled. 0 SPI frames has priority over DSI frames. 1 DSI frames have priority over SPI frames.
12 CSE	Command Select Enable

Table continues on the next page...

DSPI_DSICR1 field descriptions (continued)

Field	Description
	When the DSICR0[ITSB] is set, the CSE bit controls insertion of command selection bit (Logic 1) in the beginning of the SPI (command) frame. 0 No selection bit is inserted in the beginning of the command frame. 1 Command selection bit is inserted at the beginning of the command frame. Total number of bits in the command frame is increased by 1.
13 DSI64E	DSI 64-bit Mode Enable Enables DSI 64-bit mode, which allows the serialization and deserialization of data frames up to 64 bits in size. The following extended registers are valid only when this mode is enabled: SDR1, ASDR1, COMPR1, DDR1, SSR1, DIMR1, DPIR1, PISR4-7. 0 DSI 64-bit Mode Disabled. DSI Mode operates in the default 32 bit configuration. 1 DSI 64-bit Mode Enabled. DSI Mode operates in 64-bit configuration.
14 DSE1	Data Select Enable 1 When the TBSC bit is set, the DSE1 bit controls insertion of the zero bit (Data Select) in the middle of the data frame. The insertion bit position is defined by the FMSZ field of CTARn register, selected by the DSICR0[DSICTAS] field. If DSI 64-bit mode is enabled (DSICR1[DSI64]), then the insertion bit position is defined by the concatenation of {DSICR0[FMSZ4], CTAR[FMSZ]}. The TSBCNT field value must be greater than the position of insertion for proper operation of DSE1. 0 No Zero bit is inserted in the middle of the data frame. 1 Zero bit is inserted at the middle of the data frame. Total number of bits in the data frame is increased by 1.
15 DSE0	Data Select Enable 0 When the TBSC bit is set, the DSE1 bit controls insertion of the zero bit (Data Select) in the beginning of the DSI frame. 0 No Zero bit is inserted in the beginning of the frame. 1 Zero bit is inserted at the beginning of the data frame. Total number of bits in the data frame is increased by 1.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 DPCS1_x	DSI Peripheral Chip Select 0–7 Specifies the PCSs to assert for the second part of the DSI frame when operating in TSB or ITSB configuration with dual receiver. DPCS1 selects the PCS signals to assert during the second part of the DSI frame. DPCS1 controls the assertions of PCS signals only in TSB and ITSB mode. 0 Negate PCS[x]. 1 Assert PCS[x].

42.4.18 DSI Serialization Source Select Register 0 (DSPI_SSR0)

SSR0 is used to create a combined frame for transmission that contains bits from ASDR0 and SDR0. Each bit in the SSR0 register selects a corresponding bit to be serialized. When DSICR0[TXSS] is set, the SSR0 register value has no effect. When DSI 64-bit mode is enabled (DSICR1[DSI64E]), the concatenation of {SSR1, SSR0} provides the 64-bit Serialization Source Select Value.

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_SSR0 field descriptions

Field	Description
0–31 SS	Source Select Select the serialization source for the 32 LSB of the DSI frame. Each SS bit selects data for a corresponded bit in the transmitted frame. 0 The bit in the transmitted frame is taken from the Parallel Input pin. 1 The bit in the transmitted frame is taken from the ASDR0 register

42.4.19 DSI Parallel Input Select Register 0 (DSPI_PISR0)

PISR0–3 select each data bit for the transmitted frame from 32 parallel input pins. Each Input Pin Select (IPS) field controls one bit in the transmitted frame. Each register contains control fields for 8 bits in the frame. The select field value is defined as a 4-bit signed integer number. The selected parallel input pin number is defined as a difference between the sum of the field number and the field value.

For example, if IPS16 is equal to the binary number 1111 (minus 1 decimal), then bit 16 in the frame is taken from parallel input pin number 15. When the IPS0 is equal to -1, then bit 0 in the frame is taken from parallel input 31. If DSI 64-bit is enabled (DSICR1[DSI64E]), bit 0 is taken from parallel input 63. When the IPS0 is equal to +1, then bit 0 in the frame is taken from Parallel Input 1 and so on.

Note that PISR0–3 only preselect the Parallel Input pins. The final selection to the transmitted frame is made with the SSR0-1 register bits or the DSICR0[TXSS] bit.

When DSI 64-bit mode is enabled (DSICR1[DSI64E]), PISR4-7 handle the 32 most-significant pins of the 64 parallel input pins.

Address: 0h base + D8h offset = D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS7				IPS6				IPS5				IPS4				IPS3				IPS2				IPS1				IPS0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_PISR0 field descriptions

Field	Description
0–3 IPS7	Input Pin Select 7 Selects the Parallel Input pin for transmitted frame bit 7.
4–7 IPS6	Input Pin Select 6 Selects the Parallel Input pin for transmitted frame bit 6.
8–11 IPS5	Input Pin Select 5 Selects the Parallel Input pin for transmitted frame bit 5.
12–15 IPS4	Input Pin Select 5 Selects the Parallel Input pin for transmitted frame bit 4.
16–19 IPS3	Input Pin Select 3 Selects the Parallel Input pin for transmitted frame bit 3.
20–23 IPS2	Input Pin Select 2 Selects the Parallel Input pin for transmitted frame bit 2.
24–27 IPS1	Input Pin Select 1 Selects the Parallel Input pin for transmitted frame bit 1.
28–31 IPS0	Input Pin Select 0 Selects the Parallel Input pin for transmitted frame bit 0.

42.4.20 DSI Parallel Input Select Register 1 (DSPI_PISR1)

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS15				IPS14				IPS13				IPS12				IPS11				IPS10				IPS9				IPS8			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_PISR1 field descriptions

Field	Description
0–3 IPS15	Input Pin Select 15

Table continues on the next page...

DSPI_PISR1 field descriptions (continued)

Field	Description
	Selects the Parallel Input pin for transmitted frame bit 15.
4–7 IPS14	Input Pin Select 14 Selects the Parallel Input pin for transmitted frame bit 14.
8–11 IPS13	Input Pin Select 13 Selects the Parallel Input pin for transmitted frame bit 13.
12–15 IPS12	Input Pin Select 12 Selects the Parallel Input pin for transmitted frame bit 12.
16–19 IPS11	Input Pin Select 11 Selects the Parallel Input pin for transmitted frame bit 11.
20–23 IPS10	Input Pin Select 10 Selects the Parallel Input pin for transmitted frame bit 10.
24–27 IPS9	Input Pin Select 9 Selects the Parallel Input pin for transmitted frame bit 9.
28–31 IPS8	Input Pin Select 8 Selects the Parallel Input pin for transmitted frame bit 8.

42.4.21 DSI Parallel Input Select Register 2 (DSPI_PISR2)

Address: 0h base + E0h offset = E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS23				IPS22				IPS21				IPS20				IPS19				IPS18				IPS17				IPS16			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_PISR2 field descriptions

Field	Description
0–3 IPS23	Input Pin Select 23 Selects the Parallel Input pin for transmitted frame bit 23.
4–7 IPS22	Input Pin Select 22 Selects the Parallel Input pin for transmitted frame bit 22.
8–11 IPS21	Input Pin Select 21 Selects the Parallel Input pin for transmitted frame bit 21.
12–15 IPS20	Input Pin Select 20 Selects the Parallel Input pin for transmitted frame bit 20.

Table continues on the next page...

DSPI_PISR2 field descriptions (continued)

Field	Description
16–19 IPS19	Input Pin Select 19 Selects the Parallel Input pin for transmitted frame bit 19.
20–23 IPS18	Input Pin Select 18 Selects the Parallel Input pin for transmitted frame bit 18.
24–27 IPS17	Input Pin Select 17 Selects the Parallel Input pin for transmitted frame bit 17.
28–31 IPS16	Input Pin Select 16 Selects the Parallel Input pin for transmitted frame bit 16.

42.4.22 DSI Parallel Input Select Register 3 (DSPI_PISR3)

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS31				IPS30				IPS29				IPS28				IPS27				IPS26				IPS25				IPS24			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

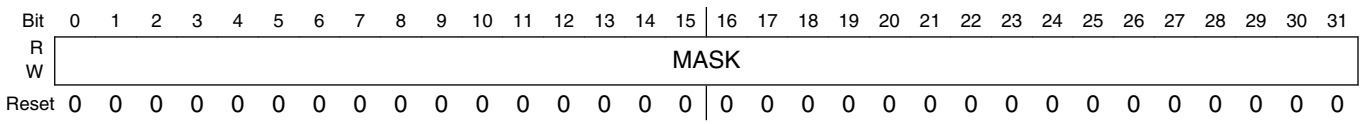
DSPI_PISR3 field descriptions

Field	Description
0–3 IPS31	Input Pin Select 31 Selects the Parallel Input pin for transmitted frame bit 31.
4–7 IPS30	Input Pin Select 30 Selects the Parallel Input pin for transmitted frame bit 30.
8–11 IPS29	Input Pin Select 29 Selects the Parallel Input pin for transmitted frame bit 29.
12–15 IPS28	Input Pin Select 28 Selects the Parallel Input pin for transmitted frame bit 28.
16–19 IPS27	Input Pin Select 27 Selects the Parallel Input pin for transmitted frame bit 27.
20–23 IPS26	Input Pin Select 26 Selects the Parallel Input pin for transmitted frame bit 26.
24–27 IPS25	Input Pin Select 25 Selects the Parallel Input pin for transmitted frame bit 25.
28–31 IPS24	Input Pin Select 24 Selects the Parallel Input pin for transmitted frame bit 24.

42.4.23 DSI Deserialized Data Interrupt Mask Register 0 (DSPI_DIMR0)

DIMR0 selects bits in the 32 least-significant bits of the received DSI frame to be checked to generate the DDI interrupt. When DSI 64-bit mode is enabled (DSICR1[DSI64E]), the concatenation of {DIMR1, DIMR0} provides the 64-bit Deserialized Data Interrupt Mask value.

Address: 0h base + E8h offset = E8h



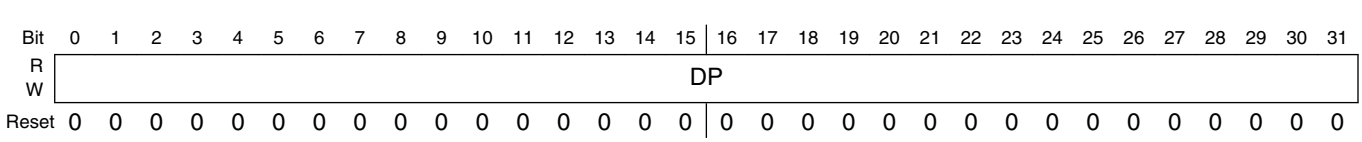
DSPI_DIMR0 field descriptions

Field	Description
0–31 MASK	Mask Defines which bits in the 32 LSB of received deserialization data should be checked to produce the Deserialized Data Interrupt (DDI). 0 The bit in the received DSI frame does not produce a DDI interrupt. 1 The bit in the received DSI frame can produce a DDI interrupt if the data bit matches the configured polarity.

42.4.24 DSI Deserialized Data Polarity Interrupt Register 0 (DSPI_DPIR0)

DPIR0 defines which data bit value in the 32 least-significant bits of the received DSI frame generates the DDI interrupt. When DSI 64-bit mode is enabled (DSICR1[DSI64E]), the concatenation of {DPIR1, DPIR0} provides the 64-bit Deserialized Data Polarity Interrupt value.

Address: 0h base + ECh offset = ECh



DSPI_DPIR0 field descriptions

Field	Description
0–31 DP	Data Polarity Defines which value of the 32 LSB of received deserialization data sets the SR[DDIF] bit. 0 If the received bit is 0, the SR[DDIF] bit is set. 1 If the received bit is 1, the SR[DDIF] bit is set.

42.4.25 DSI Serialization Data Register 1 (DSPI_SDR1)

SDR1 contains the states of the 32 MSB parallel input signals. The states of these parallel input signals are latched into the SDR1 on the rising edge of every bus clock. The SDR1 is read-only. When the TXSS bit in the DSICR0 is cleared, the data in the SDR1 and SDR0 is used as the source of the DSI frames. The concatenation of {SDR1, SDR0} provides the 64-bit data to be serialized. This register is valid only when DSI 64-bit mode is enabled (DSICR1[DSI64E]).

Address: 0h base + F0h offset = F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SER_DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_SDR1 field descriptions

Field	Description
0–31 SER_DATA	Serialized Data Contains the 32 MSB signal states of the 64 parallel input signals. This field is valid only when DSICR1[DSI64E] is set. The concatenation of {SDR1, SDR0} provides the 64-bit data to be serialized.

42.4.26 DSI Alternate Serialization Data Register 1 (DSPI_ASDR1)

ASDR1 is used by host software to write the 32 MSB of the 64-bit data to be serialized. When the TXSS bit in the DSICR0 is set, the data in the ASDR1 and ASDR0 is the source of the DSI frames. Writes to the ASDR1 take effect on the next frame boundary. The concatenation of {ASDR1, ASDR0} provides the 64-bit alternate serialization data. This register is valid only when DSI 64-bit mode is enabled (DSICR1[DSI64E]).

Address: 0h base + F4h offset = F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ASER_DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_ASDR1 field descriptions

Field	Description
0–31 ASER_DATA	Alternate Serialized Data Holds the 32 MSB of the 64-bit alternate data to be serialized. This field is valid only when DSICR1[DSI64E] is set. The concatenation of {ASDR1, ASDR0} provides the 64-bit alternate serialization data.

42.4.27 DSI Transmit Comparison Register 1 (DSPI_COMPR1)

COMPR1 holds a copy of the 32 MSB of the last transmitted 64-bit DSI data. COMPR1 is read-only. Upper 32 bits of the 64-bit DSI data is transferred to this register as it is loaded into the TX Shift Register. The concatenation of {COMPR1, COMPR0} provides the 64-bit transmit comparison data. This register is valid only when DSI 64-bit mode is enabled (DSICR1[DSI64E]).

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COMP_DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_COMPR1 field descriptions

Field	Description
0–31 COMP_DATA	Compare Data Holds the 32 MSB of the last serialized 64-bit DSI data. This field is valid only when DSICR1[DSI64E] is set. The concatenation of {COMPR1, COMPR0} provides the last serialized 64-bit DSI data.

42.4.28 DSI Deserialization Data Register 1 (DSPI_DDR1)

DDR1 holds the 32 MSB signal states for the 64 Parallel Output signals. The DDR1 is read-only and host software can read data from incoming DSI frames. The concatenation of {DDR1, DDR0} provides the 64-bit Deserialized data. This register is valid only when DSI 64-bit mode is enabled (DSICR1[DSI64E]).

Address: 0h base + FCh offset = FCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DESER_DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DDR1 field descriptions

Field	Description
0–31 DESER_DATA	Deserialized Data Holds 32 MSB of the deserialized data that is presented as signal states to the 64 Parallel Output signals.This field is valid only when DSICR1[DSI64E] is set. The concatenation of {DDR1, DDR0} provides the 64-bit deserialized data.

42.4.29 DSI Serialization Source Select Register 1 (DSPI_SSR1)

SSR1 is used to create a combined frame for transmission that contains bits from ASDR1 and SDR1. Each bit in the SSR1 register selects a corresponding bit to be serialized. When DSICR0[TXSS] is set, the SSR1 register value has no effect. The concatenation of {SSR1, SSR0} provides the 64-bit Serialization Source Select Value. This register is valid only when DSI 64-bit mode is enabled (DSICR1[DSI64E]).

Address: 0h base + 100h offset = 100h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_SSR1 field descriptions

Field	Description
0–31 SS	Source Select Select the serialization source for the 32 MSB of the 64-bit DSI frame. Each SS bit selects data for a corresponded bit in the transmitted frame. This field is valid only when DSICR1[DSI64E] is set. The concatenation of {SSR1, SSR0} provides the 64-bit serialization source select value. 0 The bit in the transmitted frame is taken from the Parallel Input pin. 1 The bit in the transmitted frame is taken from the ASDR0/1 register

42.4.30 DSI Parallel Input Select Register 4 (DSPI_PISR4)

PISR4–7 are used to select each data bit for the transmitted frame from 32 parallel input pins. Each Input Pin Select (IPS) field controls one bit in the transmitted frame. Each register contains control fields for 8 bits in the frame. The select field value is defined as a 4-bit signed integer number. The selected parallel input pin number is defined as a difference between the sum of the field number and the field value.

For example, if IPS16 is equal to the binary number 1111 (minus 1 decimal), then bit 16 in the frame is taken from parallel input pin number 15. When the IPS0 is equal to -1, then bit 0 in the frame is taken from parallel input 31 (bit 0 is taken from parallel input 63 when DSICR1[DSI64E] is enabled). When the IPS0 is equal to +1, then bit 0 in the frame is taken from parallel input 1 and so on.

Note that PISR0–7 only preselect the parallel input pins. The final selection to the transmitted frame is made with the SSR0 and SSR1 register bits or the DSICR0[TXSS] bit.

When DSI 64-bit mode is enabled (DSICR1[DSI64E]), PISR4-7 are used to handle the extra 32 bits (32 MSB pins of the 64 parallel input pins). If DSI 64-bit mode is disabled, then PISR4-7 are not used.

Address: 0h base + 104h offset = 104h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS39				IPS38				IPS37				IPS36				IPS35				IPS34				IPS33				IPS32			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_PISR4 field descriptions

Field	Description
0–3 IPS39	Input Pin Select 39 Selects the Parallel Input pin for transmitted frame bit 39.
4–7 IPS38	Input Pin Select 38 Selects the Parallel Input pin for transmitted frame bit 38.
8–11 IPS37	Input Pin Select 37 Selects the Parallel Input pin for transmitted frame bit 37.
12–15 IPS36	Input Pin Select 36 Selects the Parallel Input pin for transmitted frame bit 36.
16–19 IPS35	Input Pin Select 35 Selects the Parallel Input pin for transmitted frame bit 35.
20–23 IPS34	Input Pin Select 34 Selects the Parallel Input pin for transmitted frame bit 34.
24–27 IPS33	Input Pin Select 33 Selects the Parallel Input pin for transmitted frame bit 33.
28–31 IPS32	Input Pin Select 32 Selects the Parallel Input pin for transmitted frame bit 32.

42.4.31 DSI Parallel Input Select Register 5 (DSPI_PISR5)

Address: 0h base + 108h offset = 108h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS47				IPS46				IPS45				IPS44				IPS43				IPS42				IPS41				IPS40			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_PISR5 field descriptions

Field	Description
0–3 IPS47	Input Pin Select 47 Selects the Parallel Input pin for transmitted frame bit 47.
4–7 IPS46	Input Pin Select 46 Selects the Parallel Input pin for transmitted frame bit 46.
8–11 IPS45	Input Pin Select 45 Selects the Parallel Input pin for transmitted frame bit 45.
12–15 IPS44	Input Pin Select 44 Selects the Parallel Input pin for transmitted frame bit 44.
16–19 IPS43	Input Pin Select 43 Selects the Parallel Input pin for transmitted frame bit 43.
20–23 IPS42	Input Pin Select 42 Selects the Parallel Input pin for transmitted frame bit 42.
24–27 IPS41	Input Pin Select 41 Selects the Parallel Input pin for transmitted frame bit 41.
28–31 IPS40	Input Pin Select 40 Selects the Parallel Input pin for transmitted frame bit 40.

42.4.32 DSI Parallel Input Select Register 6 (DSPI_PISR6)

Address: 0h base + 10Ch offset = 10Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS55				IPS54				IPS53				IPS52				IPS51				IPS50				IPS49				IPS48			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_PISR6 field descriptions

Field	Description
0–3 IPS55	Input Pin Select 55 Selects the Parallel Input pin for transmitted frame bit 55.
4–7 IPS54	Input Pin Select 54 Selects the Parallel Input pin for transmitted frame bit 54.
8–11 IPS53	Input Pin Select 53 Selects the Parallel Input pin for transmitted frame bit 53.
12–15 IPS52	Input Pin Select 52

Table continues on the next page...

DSPI_PISR6 field descriptions (continued)

Field	Description
	Selects the Parallel Input pin for transmitted frame bit 52.
16–19 IPS51	Input Pin Select 51 Selects the Parallel Input pin for transmitted frame bit 51.
20–23 IPS50	Input Pin Select 50 Selects the Parallel Input pin for transmitted frame bit 50.
24–27 IPS49	Input Pin Select 49 Selects the Parallel Input pin for transmitted frame bit 49.
28–31 IPS48	Input Pin Select 48 Selects the Parallel Input pin for transmitted frame bit 48.

42.4.33 DSI Parallel Input Select Register 7 (DSPI_PISR7)

Address: 0h base + 110h offset = 110h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IPS63				IPS62				IPS61				IPS60				IPS59				IPS58				IPS57				IPS56			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_PISR7 field descriptions

Field	Description
0–3 IPS63	Input Pin Select 63 Selects the Parallel Input pin for transmitted frame bit 63.
4–7 IPS62	Input Pin Select 62 Selects the Parallel Input pin for transmitted frame bit 62.
8–11 IPS61	Input Pin Select 61 Selects the Parallel Input pin for transmitted frame bit 61.
12–15 IPS60	Input Pin Select 60 Selects the Parallel Input pin for transmitted frame bit 60.
16–19 IPS59	Input Pin Select 59 Selects the Parallel Input pin for transmitted frame bit 59.
20–23 IPS58	Input Pin Select 58 Selects the Parallel Input pin for transmitted frame bit 58.
24–27 IPS57	Input Pin Select 57 Selects the Parallel Input pin for transmitted frame bit 57.

Table continues on the next page...

DSPI_PISR7 field descriptions (continued)

Field	Description
28–31 IPS56	Input Pin Select 56 Selects the Parallel Input pin for transmitted frame bit 56.

42.4.34 DSI Deserialized Data Interrupt Mask Register 1 (DSPI_DIMR1)

DIMR1 selects bits from the 32 MSB in the received 64-bit DSI frame to be checked to generate the DDI interrupt. The concatenation of {DIMR1, DIMR0} provides the 64-bit Deserialized Data Interrupt Mask Value. This register is valid only when DSI 64-bit mode is enabled (DSICR1[DSI64E]).

Address: 0h base + 114h offset = 114h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DIMR1 field descriptions

Field	Description
0–31 MASK	Mask Defines which of the 32 MSB in the received 64-bit deserialization data should be checked to produce the Deserialized Data Interrupt (DDI). 0 The bit in the received DSI frame does not produce a DDI interrupt. 1 The bit in the received DSI frame can produce a DDI interrupt if the data bit matches the configured polarity.

42.4.35 DSI Deserialized Data Polarity Interrupt Register 1 (DSPI_DPIR1)

DPIR1 defines which of the 32 MSB value in the received 64-bit DSI frame generates the DDI interrupt. The concatenation of {DPIR1, DPIR0} provides the 64-bit Deserialized Data Polarity Value. This register is valid only when DSI 64-bit mode is enabled (DSICR1[DSI64E]).

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_DPIR1 field descriptions

Field	Description
0–31 DP	Data Polarity Defines which of the 32 MSB of the received 64-bit deserialization data sets the SR[DDIF] bit. 0 If the received bit is 0, the SR[DDIF] bit is set. 1 If the received bit is 1, the SR[DDIF] bit is set.

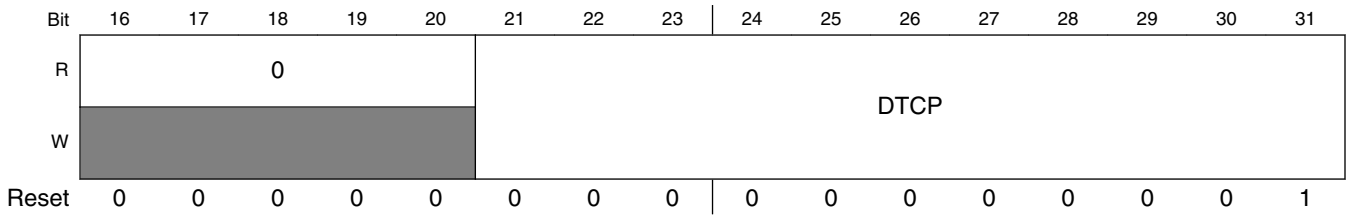
42.4.36 Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*)

CTARE registers are used to define the extended transfer attributes for an SPI frame. These registers are valid only when Extended SPI mode is enabled (MCR[XSPI]).

When the module is configured as a SPI master, the CTAS field in CMD FIFO entry selects which of the CTARE registers is used.

Address: 0h base + 11Ch offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															FMSZE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



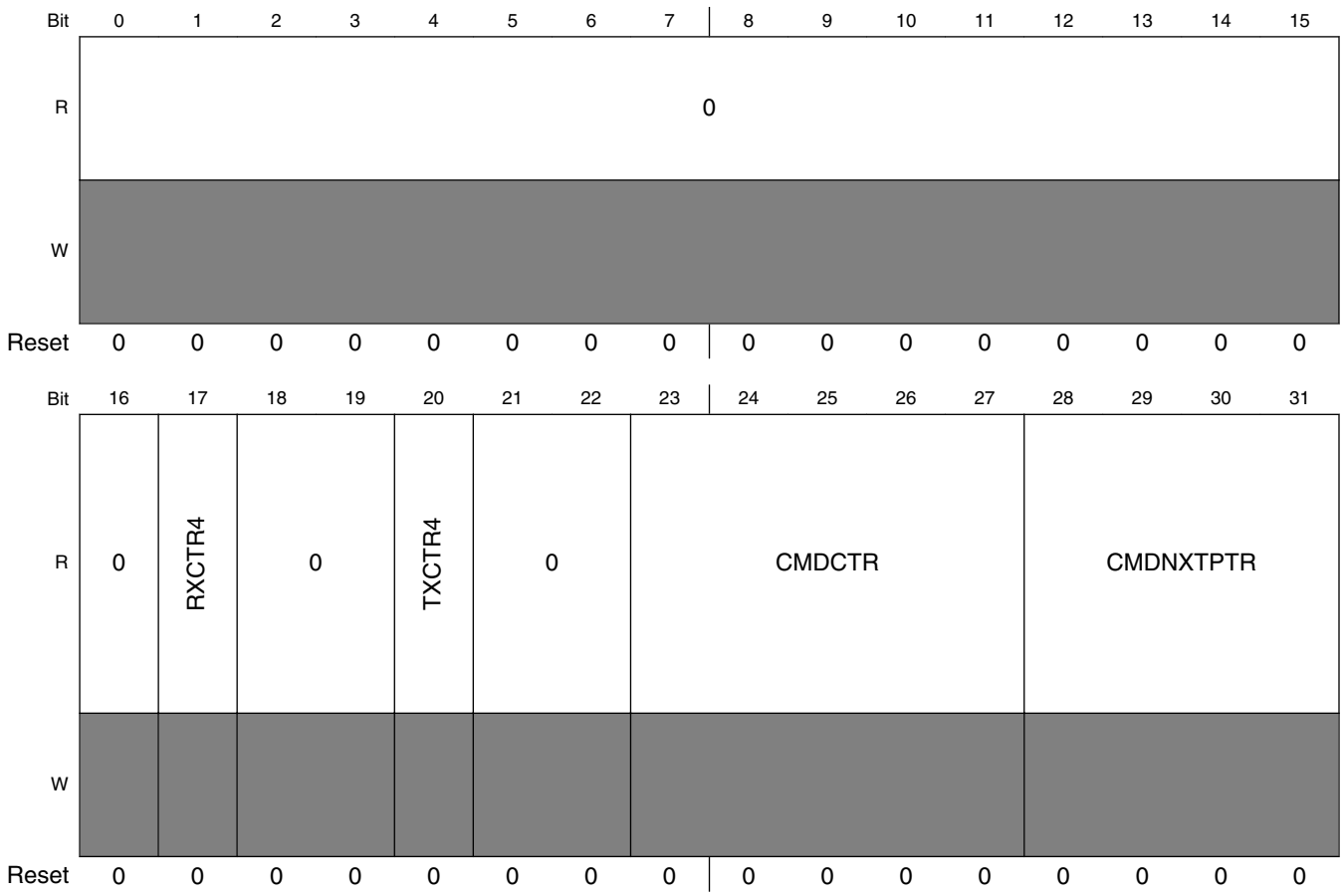
DSPI_CTARE*n* field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 FMSZE	Frame Size Extended This field is valid only when MCR[XSPI] is set. This field concatenated with CTAR[FMSZ] defines the Frame size of the SPI frames to be transmitted. Effective frame size would be the concatenation of {CTARE[FMSZE], CTAR[FMSZ]} plus 1. 0 Default Mode. Up to 16 bit SPI frames can be transferred. 1 Up to 32 bit SPI frames can be transferred. Each Frame transfer will be a result of 2 simulataneous TX FIFO Pop Operation.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–31 DTCP	Data Transfer Count Preload This field is valid only when DSPIx_MCR[XSPI] is set. This field defines the number of data frames (whose size is defined by CTARE[FMSZE] and CTAR[FMSZ]) to be transmitted using the Command frame which selected this DSPIx_CTARE register. The value 0 is reserved and should not be written in this field. The default value of this field is 1.

42.4.37 Status Register Extended (DSPI_SREX)

The register contains status fields. The fields reflect the status of the module and indicate the occurrence of events. This register is not writable.

Address: 0h base + 13Ch offset = 13Ch



DSPI_SREX field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 RXCTR4	RX FIFO Counter[4] This bit is an extension of SR[RXCTR]. The concatenated field {RXCTR4, RXCTR} indicates the number of entries in the RX FIFO. This field is decremented every time the POPR is read. And this field is incremented every time data is transferred from the shift register to the RX FIFO.
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

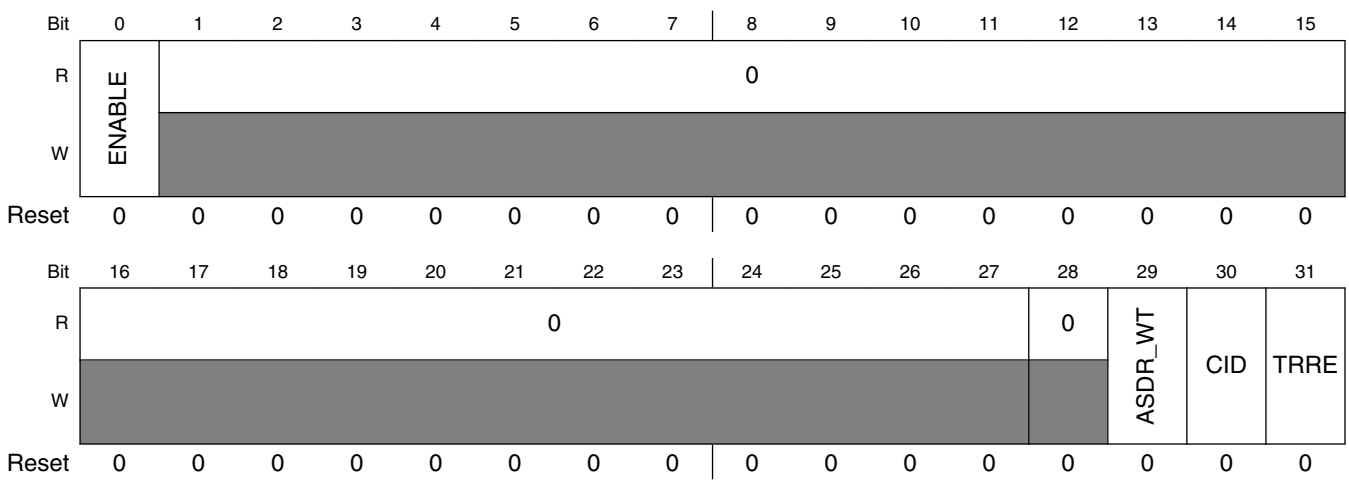
DSPI_SREX field descriptions (continued)

Field	Description
20 TXCTR4	TX FIFO Counter[4] This bit is an extension of SR[TXCTR]. The concatenated field {TXCTR4, TXCTR} indicates the number of valid entries in the TX FIFO. This field is incremented every time the PUSHR is written. And this field is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
21–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–27 CMDCTR	CMD FIFO Counter Indicates the number of entries in the CMD FIFO. The CMDCTR is incremented every time the command part of PUSHR is written. The CMDCTR is decremented every time a SPI command is executed (all data frames due to current command frame have been transmitted).
28–31 CMDNXTPTR	Command Next Pointer Indicates which CMD FIFO Entry is used during the next transfer. The CMDNXTPTR field is updated every time SPI data due to current command have been transmitted.

42.4.38 Trigger Register (DSPI_TRIG)

This register provides the user to configure trigger settings for various modes. The trigger settings in this register are activated only if the bit TRIG[ENABLE] is set.

Address: 0h base + 140h offset = 140h



DSPI_TRIG field descriptions

Field	Description
0 ENABLE	Trigger Register Enable This bit is used to enable the TRIG register.

Table continues on the next page...

DSPI_TRIG field descriptions (continued)

Field	Description
	0 TRIG register is disabled. Trigger settings configured in this register have no effect on the module. 1 TRIG register is enabled. Trigger settings configured in this register are valid.
1–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 ASDR_WT	ASDR Write Trigger Enable Enables the module to initiate a DSI frames transfer on a write operation performed on ASDR0 register. 0 ASDR0 write trigger disabled. 1 ASDR0 write trigger enabled.
30 CID	Change In Data Transfer Enable Enables a change in serialization data to initiate a DSI frames transfer in DSI and CSI configurations. When the CID bit is set, DSI frames are initiated when the current DSI data differs from the previous DSI data shifted out. **NOTE:** When TRIG[ENABLE] is set, the CID bit is given by TRIG[CID] which overrides DSICR0[CID].
31 TRRE	Trigger Reception Enable Enables the module to initiate a DSI frames transfer with an external trigger signal. **NOTE:** When TRIG[ENABLE] is set, the TRRE bit is given by TRIG[TRRE] which overrides DSICR0[TRRE]. 0 Trigger signal reception disabled. 1 Trigger signal reception enabled.

42.4.39 Time Slot Length Register (DSPI_TSL)

The TSL register is only valid when DSICR0[ITSB] is enabled and internal triggering mechanism is used. It provides the length of every time slot in terms of baud cycles.

Address: 0h base + 150h offset = 150h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	TS3_LEN							0	TS2_LEN						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TS1_LEN							0	TS0_LEN						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_TSL field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 TS3_LEN	Time Slot 3 Length This field provides the length of TS3 in terms of baud cycles. It must be ensured that the programmed value is greater than or equal to the length of the frame to be transmitted during TS3.Time Slot3 Length = TS3_LEN+1.This time slot shall be skipped if programmed to 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 TS2_LEN	Time Slot 2 Length This field provides the length of TS2 in terms of baud cycles. It must be ensured that the programmed value is greater than or equal to the length of the frame to be transmitted during TS2.Time Slot 2 Length = TS2_LEN+1.This time slot shall be skipped if programmed to 0.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–23 TS1_LEN	Time Slot 1 Length This field provides the length of TS1 in terms of baud cycles. It must be ensured that the programmed value is greater than or equal to the length of the frame to be transmitted during TS1.Time Slot 1 Length = TS1_LEN+1.This time slot shall be skipped if programmed to 0.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 TS0_LEN	Time Slot 0 Length This field provides the length of TS0 in terms of baud cycles. It must be ensured that the programmed value is greater than or equal to the length of the frame to be transmitted during TS0.Time Slot 0 Length = TS0_LEN+1.This time slot shall be skipped if programmed to 0.

42.4.40 Time Slot Configuration Register (DSPI_TS_CONF)

This register provides the basic configuration options available for the triggered implementation of MSC protocol. This register is only valid when DSICR0[ITSB] is enabled. Some configurations in this register check for the validity of command frame. The following conditions must be true for a command frame to be valid : a) The previous transmitted frame should not be a command frame. b) Data required to transmit the command frame should be available.

Address: 0h base + 154h offset = 154h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TS3				TS2				TS1				TS0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DSPI_TS_CONF field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–19 TS3	Time Slot 3 This field is used to program the frame rule to be followed during Time Slot 3. This field is only valid when DSICR0[ITSB] mode is enabled. 0000 PF - Passive Frame 0001 SRDF - Single Receiver Data Frame 0010 DRF1 - First part of Dual Receiver Frame 0011 DRF2 - Second part of Dual Receiver Frame 0100 CMD_PF - Command Frame if valid, else Passive Frame 0101 CMD_SRDF - Command Frame if valid, else Single Receiver Data Frame 0110 CMD_DRF1 - Command Frame if valid, else first half of Dual Receiver Data Frame 0111 CMD_DRF2 - Command Frame if valid, else second half of Dual Receiver Data Frame 1xxx Reserved
20–23 TS2	Time Slot 2 This field is used to program the frame rule to be followed during Time Slot 2. This field is only valid when DSICR0[ITSB] mode is enabled. 0000 PF - Passive Frame 0001 SRDF - Single Receiver Data Frame 0010 DRF1 - First part of Dual Receiver Frame 0011 DRF2 - Second part of Dual Receiver Frame 0100 CMD_PF - Command Frame if valid, else Passive Frame 0101 CMD_SRDF - Command Frame if valid, else Single Receiver Data Frame 0110 CMD_DRF1 - Command Frame if valid, else first half of Dual Receiver Data Frame 0111 CMD_DRF2 - Command Frame if valid, else second half of Dual Receiver Data Frame 1xxx Reserved
24–27 TS1	Time Slot 1 This field is used to program the frame rule to be followed during Time Slot 1. This field is only valid when DSICR0[ITSB] mode is enabled. 0000 PF - Passive Frame 0001 SRDF - Single Receiver Data Frame 0010 DRF1 - First part of Dual Receiver Frame 0011 DRF2 - Second part of Dual Receiver Frame 0100 CMD_PF - Command Frame if valid, else Passive Frame 0101 CMD_SRDF - Command Frame if valid, else Single Receiver Data Frame 0110 CMD_DRF1 - Command Frame if valid, else first half of Dual Receiver Data Frame 0111 CMD_DRF2 - Command Frame if valid, else second half of Dual Receiver Data Frame 1xxx Reserved
28–31 TS0	Time Slot 0 This field is used to program the frame rule to be followed during Time Slot 0. This field is only valid when DSICR0[ITSB] mode is enabled. 0000 PF - Passive Frame

Table continues on the next page...

DSPI_TS_CONF field descriptions (continued)

Field	Description
	0001 SRDF - Single Receiver Data Frame 0010 DRF1 - First part of Dual Receiver Frame 0011 DRF2 - Second part of Dual Receiver Frame 0100 CMD_PF - Command Frame if valid, else Passive Frame 0101 CMD_SRDF - Command Frame if valid, else Single Receiver Data Frame 0110 CMD_DRF1 - Command Frame if valid, else first half of Dual Receiver Data Frame 0111 CMD_DRF2 - Command Frame if valid, else second half of Dual Receiver Data Frame 1xxx Reserved

42.5 Functional description

The module supports full-duplex, synchronous serial communications between chips and peripheral devices. The module can also be used to reduce the number of pins required for I/O by serializing and deserializing Parallel Input/Output signals.All communications are done with SPI-like protocol.

The module has the following configurations

- The SPI Configuration in which the module operates as a basic SPI or a queued SPI.
- DSI Configuration in which the module serializes and deserializes Parallel Input/ Output signals or bits from memory-mapped registers
- When Master mode is enabled, the module also supports CSI Configuration, which combines the functionality of the SPI and DSI configurations.

The DCONF field in the Module Configuration Register (MCR) determines the module Configuration. See Module Configuration Register (DSPI_MCR) for the DSPI configuration values.

The CTARn registers hold clock and transfer attributes. The SPI configuration allows to select which CTAR to use on a frame by frame basis by setting a field in the SPI command. The Extended SPI Mode (DSPIx_MCR[XSPI]) further allows the usage CTAREn (CTARn Extended) registers which allows the user to send multiple data frames using a single command frame.

The DSI configuration statically selects which CTAR to use. In CSI, Configuration priority logic determines if SPI data or DSI data is transferred and dictates what CTAR register is used for the data transfer. See Clock and Transfer Attributes Register (In

Master Mode) (DSPI_CTAR*n*) for information on the fields of CTAR registers. See Clock and Transfer Attributes Register Extended (DSPI_CTARE*n*) for information on the fields of CTARE registers.

Typical master to slave connections are shown in the following figure. When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave. The data that was in the master shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the Transfer Control Flag(TCF) bit in the Shift Register(SR) is set to indicate a completed frame transfer.

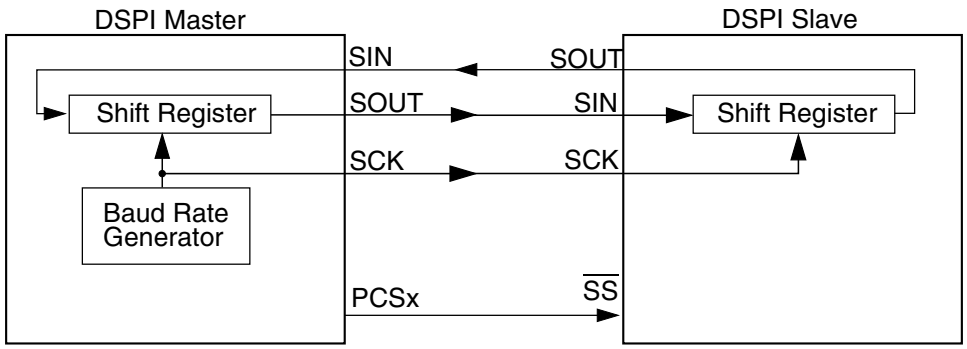


Figure 42-3. Serial protocol overview

Generally, more than one slave device can be connected to the module master. 6 Peripheral Chip Select (PCS) signals of the module masters can be used to select which of the slaves to communicate with. Refer to the chip specific section for details on the number of PCS signals used in this chip.

The three DSPI configurations share transfer protocol and timing properties which are described independently of the configuration in Transfer formats. The transfer rate and delay settings are described in Module baud rate and clock delay generation.

42.5.1 Start and Stop of module transfers

The module has two operating states: Stopped and Running. Both the states are independent of it's configuration. The default state of the module is Stopped. In the Stopped state, no serial transfers are initiated in Master mode and no transfers are responded to in Slave mode. The Stopped state is also a safe state for writing the various configuration registers of the module without causing undetermined results. In the Running state serial transfers take place.

The TXRXS bit in the SR indicates the state of module. The bit is set if the module is in Running state.

The module starts or transitions to Running when all of the following conditions are true:

- SR[EOQF] bit is clear
- Chip is not in the Debug mode or the MCR[FRZ] bit is clear
- MCR[HALT] bit is clear

The module stops or transitions from Running to Stopped after the current frame when any one of the following conditions exist:

- SR[EOQF] bit is set
- Chip in the Debug mode and the MCR[FRZ] bit is set
- MCR[HALT] bit is set

State transitions from Running to Stopped occur on the next frame boundary if a transfer is in progress, or immediately if no transfers are in progress.

42.5.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The module is in SPI configuration when the DCONF field in the MCR is 0b00. The SPI frames can be 32 bits long. The host CPU or a DMA controller transfers the SPI data from the external to the module RAM queues to a TX FIFO buffer. The received data is stored in entries in the RX FIFO buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the module. The operation of FIFO buffers is described in the following sections:

- Transmit First In First Out (TX FIFO) buffering mechanism
- Command First In First Out (CMD FIFO) Buffering Mechanism
- Receive First In First Out (RX FIFO) buffering mechanism

The interrupt and DMA request conditions are described in Interrupts/DMA requests.

The SPI configuration supports two block-specific modes—Master mode and Slave mode. In Master mode the module initiates and controls the transfer according to the fields of the executing SPI Command. In Slave mode, the module responds only to transfers initiated by a bus master external to it and the SPI command field space is reserved.

42.5.2.1 Master mode

In SPI Master mode, the module initiates the serial transfers by controlling the SCK and the PCS signals. The executing SPI Command determines which CTARs will be used to set the transfer attributes and which PCS signals to assert. The command field also contains various bits that help with queue management and transfer protocol. See PUSH TX FIFO Register In Master Mode (DSPI_PUSHR) for details on the SPI command fields. The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin. In SPI Master mode, each SPI frame to be transmitted has a command associated with it, allowing for transfer attribute control on a frame by frame basis. In Extended SPI Master Mode, multiple SPI frames can have a single command associated with them allowing for efficient SPI frame transfers requiring common transfer attributes. In addition, the Extended SPI Mode allows for larger frame sizes of up to 32 bits.

42.5.2.2 Slave mode

In SPI Slave mode the module responds to transfers initiated by an SPI bus master. It does not initiate transfers. Certain transfer attributes such as clock polarity, clock phase, and frame size must be set for successful communication with an SPI master. The SPI Slave mode transfer attributes are set in the CTAR0 and CTARE0. The data is shifted out with MSB first. Shifting out of LSB is not supported in this mode.

42.5.2.3 FIFO disable operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO, CMD FIFO or RX FIFO. The module operates as a double-buffered simplified SPI when the FIFOs are disabled. The Transmit and Receive side of the FIFOs are disabled separately. Setting the MCR[DIS_TXF] bit disables the TX FIFO and CMD FIFO, and setting the MCR[DIS_RXF] bit disables the RX FIFO.

The FIFO disable mechanisms are transparent to the user and to host software. Transmit data and commands are written to the PUSHR and received data is read from the POPR.

When the TX FIFO and CMD FIFO are disabled:

- SR[TFFF], SR[TFUF], SR[CMDFFF], SREX[CMDCTR] and SR[TXCTR] behave as if there is a one-entry FIFO
- The contents of TXFRs, SR[TXNXTPTR] and SREX[CMDNXTPTR] are undefined

Similarly, when the RX FIFO is disabled, the RFDF, RFOF, and RXCTR fields in the SR behave as if there is a one-entry FIFO, but the contents of the RXFR registers and POPNXTPTR are undefined.

42.5.2.4 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission. The TX FIFO holds 4 words, each consisting of SPI data. The number of entries in the TX FIFO is device-specific. SPI data is added to the TX FIFO by writing to the Data Field of module PUSH FIFO Register (PUSHR). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the module Status Register (SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time a 8- or 16-bit write takes place to PUSHR[TXDATA] or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates the TX FIFO Entry that will be transmitted during the next transfer. The TXFRn Registers are invalid in the Extended SPI Mode, since the TX FIFO and CMD FIFO can be used independently. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR number and it rolls over after reaching the maximum.

42.5.2.4.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO and CMD FIFO by writing to the PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set. The TFFF bit is cleared when TX FIFO is full and the DMA controller indicates that a write to PUSHR is complete. Writing a '1' to the TFFF bit also clears it. The TFFF can generate a DMA request or an interrupt request. See Transmit FIFO Fill Interrupt or DMA Request for details.

The module ignores attempts to push data to a full TX FIFO, and the state of the TX FIFO does not change and no error condition is indicated.

42.5.2.4.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter decrements by one. When Extended SPI Mode (DSPIx_MCR[XSPI]) is enabled, if the frame size of SPI Data to be transmitted is more than 16 bits, then it causes two Data entries to be popped from TX FIFO simultaneously which are transferred to the shift register. The first of the two popped entries forms the 16 least significant bits of the SPI frame to be transmitted. Such

an operation also causes TX FIFO Counter to decrement by two. At the end of a transfer, the TCF bit in the SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a '1' to the CLR_TXF bit in MCR.

If an external bus master initiates a transfer with a module slave while the slave's TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave's SR is set. See Transmit FIFO Underflow Interrupt Request for details.

42.5.2.5 Command First In First Out (CMD FIFO) Buffering Mechanism

The CMD FIFO functions as a buffer of SPI command used for SPI data transmission. When Extended SPI Mode (MCR[XSPI]) is disabled, the TX FIFO and CMD FIFO must be filled together, i.e. write enables should be given for both the Data and Command fields while performing a PUSHR operation.When Extended SPI Mode (MCR[XSPI]) is enabled, the TX FIFO and CMD FIFO can be filled independently.

The CMD FIFO holds 4 words, each representing SPI command fields. The number of entries in the CMD FIFO is device-specific. SPI Command is added to the CMD FIFO by writing to the command field of DSPI PUSH FIFO Register (PUSHR). CMD FIFO entries can only be removed from the CMD FIFO by being shifted out (to help transmit SPI data) or by flushing the CMD FIFO.

When Extended SPI Mode (MCR[XSPI]) is disabled, every CMD FIFO entry has a corresponding single TX FIFO entry attached to it because both these FIFO's are filled simultaneously.

When Extended SPI Mode (MCR[XSPI]) is enabled, every CMD FIFO entry can have multiple TX FIFO entries attached to it. Thus a single CMD FIFO entry can be used to transmit multiple TX FIFO entries. The CTARE[DTCP] field decides the number of SPI Data Frames having frame size as {FMSZE, FMSZ} to be transmitted using the current Command Entry. The CTAR/CTARE registers pointed by the CTAS field in the Command frame gives the FMSZ and FMSZE fields respectively. The time for which a command entry is in use is known as a Command Cycle. The Busy Flag SR[BSYF] is asserted for the duration of the Command Cycle except for the last SPI frame in the Command Cycle.

The CMD FIFO Counter field (CMDCTR) in the DSPI Status Register (SR) indicates the number of valid entries in the CMD FIFO. The CMDCTR field is updated every time a 8- or 16-bit write takes place on the lower half of DSPI_PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The TXFRn Registers are invalid in the Extended SPI Mode, since the TX FIFO and CMD FIFO can be used independently. The CMDNXTPTR field indicates which CMD FIFO Entry will be used during the next command cycle. The CMDNXTPTR field is incremented every time the last SPI data in the command cycle is transferred from the TX FIFO to the shift register and it rolls over after reaching the maximum.

42.5.2.6 Receive First In First Out (RX FIFO) buffering mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds 4 received SPI data frames. The number of entries in the RX FIFO is device-specific. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data are removed (popped) from the RX FIFO by reading the module POP RX FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the module's Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the SR points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in a number of 32-bit registers. For example, POPNXTPTR equal to two means that the RXFR2 contains the received SPI data that will be returned when the POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR number and it rolls over after reaching the maximum.

42.5.2.6.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time an SPI frame is transferred to the RX FIFO, the RX FIFO Counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the SR is set indicating an overflow condition. Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

42.5.2.6.2 Draining the RX FIFO

Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the module POP RX FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data, read from the empty RX FIFO, is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete or by writing a 1 to it.

42.5.3 Deserial Serial Interface (DSI) Configuration

The DSI Configuration supports pin count reduction by serializing Parallel Input signals or register bits and shifting them out in a SPI-like protocol. The timing and transfer protocol is described in Transfer formats. The received serial frames are converted to a parallel form (deserialized) and placed on the Parallel Output signals or in the DSPI_DDR register. The various features of the DSI Configuration are set in DSPI DSI Configuration Registers (DSPI_DSICR0/1).

The DSI frames can be from 4 to 32 bits. This can go up to 64 bits when DSICR1[DSI64E] is enabled.

With Multiple Transfer Operation (MTO) the DSPI supports serial chaining of DSPI blocks within the device to create DSI frames up to 64 bits, consisting of concatenated bits from multiple DSPIs. The DSPI also supports parallel chaining allowing several DSPIs and off-chip SPI devices to share the same Serial Communications Clock (SCK) and Peripheral Chip Select (PCS) signals. See Multiple Transfer Operation (MTO) for details on the serial and parallel chaining support.

Multiple Transfer Operation (MTO) and DSICR1[DSI64E] are not supported together and cannot be enabled simultaneously.

42.5.3.1 DSI Master Mode

In DSI master mode the DSPI initiates and controls the DSI transfers. The DSI master has different conditions that can initiate a transfer:

- Continuous
- Change in data
- Trigger signal

- ASDR write
- Trigger signal combined with a change in data
- Trigger signal combined with ASDR write
- Change in data combined with ASDR write
- Trigger signal combined with change in data combined with ASDR write

These transfer initiation conditions are described in DSI Transfer Initiation Control . Transfer attributes are set during initialization. The DSICTAS field in the DSPI_DSICR0 determines which of the DSPI_CTAR registers will control the transfer attributes.

42.5.3.2 Slave Mode

In DSI slave mode the DSPI responds to transfers initiated by a SPI or DSI bus master. In this mode the DSPI does not initiate DSI transfers. Certain transfer attributes such as clock polarity and phase must be set for successful communication with a DSI master. The DSI slave mode transfer attributes are set in the DSPI_CTAR1. The data is shifted out with MSB first.

If the CID bit in the DSPI_DSICR0 is set and the data in DSPI_COMPR0/1 differs from the selected source of the serialized data, the slave DSPI will assert the $\overline{\text{MTRIG}}$ signal. If the slave's HT signal is asserted and the TRRE is set, the slave DSPI asserts $\overline{\text{MTRIG}}$. Also, if TRIG[ASDR_WT] is set and an ASDR0 write is detected, the slave DSPI will assert the $\overline{\text{MTRIG}}$ signal. These features are included to support chaining of several DSPI. Details about the $\overline{\text{MTRIG}}$ signal is found in Multiple Transfer Operation (MTO).

42.5.3.3 DSI Serialization

In the DSI Configuration, up to 64 bits can be serialized using two different sources. The TXSS bit in the DSPI_DSICR0 selects between the DSPI DSI Serialization Data Register (DSPI_SDR0–1) and the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR0–1) as the source of the serialized data. The DSPI_SDR0 holds the latest 32 Parallel Input signal values which is sampled at every rising edge of the bus clock. When DSICR1[DSI64E] is enabled, 64-bit DSI Frames are supported and hence DSPI_SDR1 holds up to 32 most significant parallel input signal values which is sampled at every rising edge of the bus clock.The DSPI_ASDR0–1 register is written by host software and used as an alternate source of serialized data. (DSPI_ASDR1 is only usable when DSICR1[DSI64E] is enabled).

The DSPI has to sample the inputs (from SDR0–1 or ASDR0–1) before it transmits a DSI frame. Due to the asynchronous clocking structure within DSPI, the sampled data (from SDR0–1 or ASDR0–1) must be synchronized into the Protocol Domain before being loaded into the Shift Register for transmission.

To prevent Clock Domain Crossing issues, inputs from SDR0–1 or ASDR0–1 should not change while they are being sampled into the Protocol Domain. This is ensured in the following manner for the ASDR0 inputs. The initial write performed on ASDR0 register will cause the 32-bit ASDR0 input to be latched into a temporary register which is then locked (further latching is prevented) until this data gets synchronized into the Protocol Domain. Once synchronized, the lock is removed and the next ASDR0 write will again trigger this process. If there was a write performed on ASDR0 while the lock was asserted, this will be taken into consideration as soon as the previous synchronization process completes and lock gets de-asserted. In case,multiple writes were performed on ASDR0 when lock was asserted, only the latest value will be considered for synchronization once the lock gets de-asserted. This process is true for ASDR1 register as well.

The entire process explained above is also true for the SDR0 and SDR1 inputs, the only difference being that SDR0–1 inputs are delayed internally by a single Bus clock (register interface clock) to detect a change in the parallel input pins.

Once the synchronization process for a particular De-serial input begins, it takes up to 5 Bus clock (register interface clock) periods and 5 Protocol clock periods before the synchronization process for the next De-serial input can begin. Hence the writes performed on ASDR0–1 and SDR0–1 are constrained with this timing.

The DSPI_PISR0-3 registers (DSPI_PISR0-7 registers if DSICR1[DSI64E] is enabled) allow to change relative position of the Parallel input pins in the transmitted frame. Each transmitted frame bit can be selected from 16 adjacent Parallel Inputs by writing IPSn fields. The IPSn field is treated as 4 bit integer number, representing numbers from -8 to 7. The Parallel Input pin number, selected by IPSn field is defined by the difference between the IPSn field number (n) and the IPSn field value. If the operation result is negative the number 32 should be added. If the result is higher than 32, 32 should be subtracted from the result. (When DSICR1[DSI64E] is enabled, if the above operation result is negative, the number 64 should be added. If the result is higher than 64, 64 should be subtracted from the result).

For example, IPS0, set to minus 1 (binary 1111), preselects Parallel Input 1 to 0 position in the transmitted frame. (This preselects parallel input to 0 position in the transmitted frame when DSICR1[DSI64E] is set). Following examples are true when DSICR1[DSI64E] is disabled.

IPS6, set to 3 (binary 0011), preselects Parallel Input 3 to be bit number 6 in the transmitted frame. The value minus 2 (1110) preselects Parallel Input 8.

IPS31, set to minus 8 (binary 1000), preselects Parallel Input 7 to be bit number 31 in the transmitted frame.

Note

The Parallel Input pin state, to be transmitted, should be selected by TXSS or DSPI_SSR0–1 register and the frame size should be higher than the bit position in the preselected frame.

DSPI_SSR0–1 register provides additional way to create the frame for transmission. Each bit from this register is OR'd with the TXSS bit and controls individual transmitted bit source. This way, the transmitted frame can have any combination of the DSPI_SDR0–1 and DSPI_ASDR0–1 bits. This feature allows control SPI based devices, requiring control and data fields in the frame. Control field may come from DSPI_ASDR0–1 register, set by the device's CPU, while data field can be generated by device peripheral modules, like PWM timers.

A copy of the last DSI frame shifted out of the Shift Register is stored in the DSPI DSI Transmit Comparison Registers (DSPI_COMPR0–1). This register provides added visibility for debugging and it serves as a reference for transfer initiation control. The following figure shows the DSI Serialization logic.

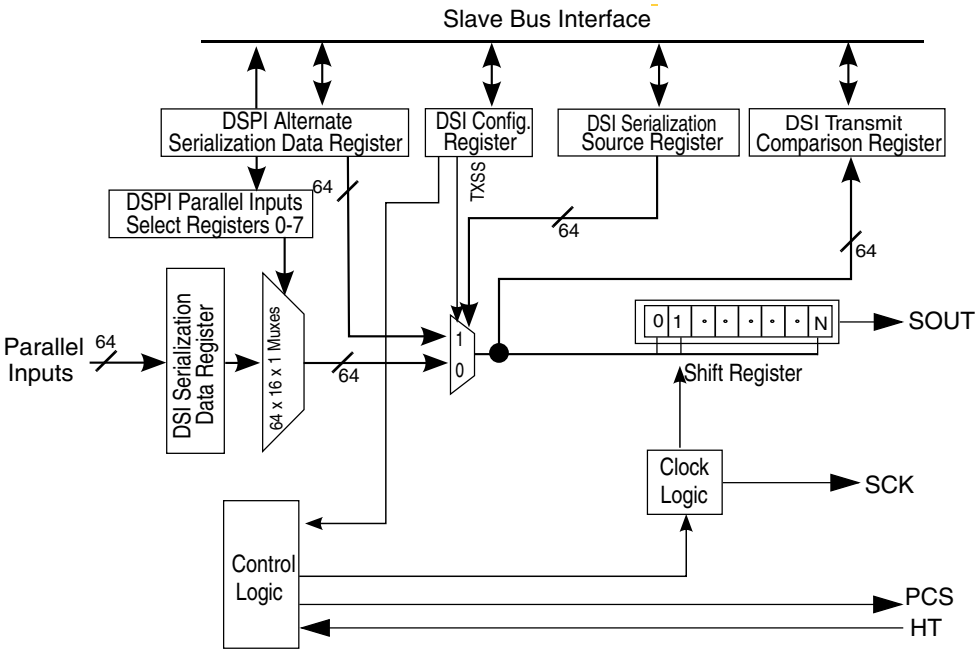


Figure 42-4. DSI Serialization Diagram

42.5.3.4 DSI Deserialization

When all bits in a DSI frame have been shifted in, the frame is copied to the DSPI DSI Deserialization Data Register (DSPI_DDR0/1). This register presents the deserialized data as Parallel Output signal values. DSPI_DDR0/1 is memory mapped to allow host software to read the deserialized data directly.

The received data is bit-wise compared to the value of the DSI Deserialized Data Polarity Interrupt Register (DSPI_DPIR0/1), bit-wise AND'ed with DSI Deserialized Interrupt Mask Register (DSPI_DIMR0/1) and the results OR'ed to produce DDIF flag in the DSPI_SR register; which in turn can cause DDI interrupt request or DMA request based on RSER[DDIF_RE] and RSER[DDIF_DIRS] bits and/or stop DSI frame transmissions if DSICR0[DMS] bit is set.

The following figure shows the DSI Deserialization logic.

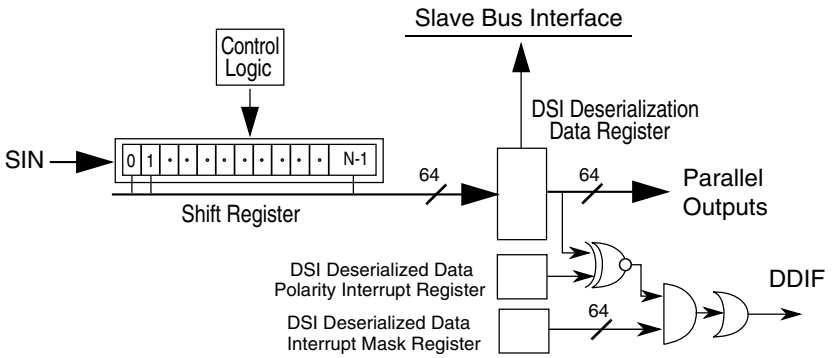


Figure 42-5. DSI Deserialization Diagram

42.5.3.5 DSI Transfer Initiation Control

Data transfers for a DSPI in DSI configuration are initiated by a condition. The transfer initiation conditions are selected by the TRRE and CID bits in the DSPI_DSICR0.

NOTE

By default, the TRRE and CID bits are programmed using DSICR0[TRRE] and DSICR0[CID]. When TRIG[ENABLE] is set, the TRRE and CID bits are given by TRIG[TRRE] and TRIG[CID] respectively and the bits DSICR0[TRRE], DSICR0[CID] have no effect.

The following table lists the transfer initiation conditions.

Table 42-8. DSI Data Transfer Initiation Control

TRRE	CID	ASDR_WT	Transfer Initiation Control
0	0	0	Continuous
0	0	1	ASDR Write
0	1	0	Change in Data
0	1	1	Change in Data or ASDR Write
1	0	0	Triggered
1	0	1	Triggered or ASDR Write
1	1	0	Triggered or Change in Data
1	1	1	Triggered or Change in Data or ASDR Write

42.5.3.5.1 Continuous Control

For Continuous Control a new DSI frame shifts out when the previous transfer cycle has completed and the Delay after Transfer (t_{DT}) has elapsed. This has no effect when DSPI is in Slave Mode.

42.5.3.5.2 ASDR Write Control

For ASDR Write Control, a DSI frame shifts out whenever a write operation to ASDR0 is detected. For ASDR_WT to have any affect, TRIG[ASDR_WT] bit must be set. Note that ASDR0 write operations will not generate ASDR write triggers when the module is in STOPPED state, including when MCR[HALT] is asserted.

NOTE

If a DSI frame > 32 bits needs to be transmitted, the user must update ASDR1 before updating ASDR0.

42.5.3.5.3 Change In Data Control

The $\overline{\text{MTRIG}}$ output signal is asserted every time a change in data is detected.

For change in data control a transfer is initiated when the data to be serialized has changed since the transfer of the last DSI frame. A copy of the previously transferred DSI data is stored in DSPI_COMPR0/1. When the data selected for the transfer from the DSPI_SDR0/1 and DSPI_ASDR0/1 registers is different from the data in DSPI_COMPR0/1 a new DSI frame is transmitted. The $\overline{\text{MTRIG}}$ output signal is asserted every time a change in data is detected.

NOTE

It is not advisable to configure a DSPI slave with the combination DSICR0[DCONT] = 1, DSICR0[CID] = 1 ; since the following statement then holds true.

Once slave select is asserted, there is a very small window (of up to t module clocks) after every SOF (Start of Frame) during which if a write on ASDR/SDR registers occur, the next frame to be transmitted would be loaded into the shift register. However, it is not feasible that such a write be initiated and completed within this short span of time; hence the data transmitted by DSPI slave would lag by one data frame.

42.5.3.5.4 Change In Data or ASDR Write Control

For Change in Data or ASDR Write Control, initiation of a transfer is controlled by the detection of a change in data to be serialized or whenever a write operation to ASDR0 is detected.

NOTE

For any DSI transfer initiation combination where TRIG[ASDR_WT] and DSICR0[HT] are both enabled, it is possible that the ASDR0 write operation causes a Change of DSI Data. This special condition results in transmission of a single DSI frame even though 2 separate triggers were generated.

42.5.3.5.5 Triggered Control

For Triggered Control initiation of a transfer is controlled by the Hardware Trigger signal (HT). The TPOL bit in the DSPI_DSICR0 selects the active edge of HT. For HT to have any affect, DSICR0[TRRE] bit must be set. Note that HT will not have any effect when the module is in STOPPED state, including when MCR[HALT] is asserted.

42.5.3.5.6 Triggered or ASDR Write Control

For Triggered or ASDR Write Control, initiation of a transfer is controlled by the HT signal or whenever a write operation to ASDR0 is detected.

42.5.3.5.7 Triggered or Change In Data Control

For Triggered or Change in Data Control, initiation of a transfer is controlled by the HT signal or by the detection of a change in data to be serialized.

42.5.3.5.8 Triggered or Change In Data or ASDR Write Control

For Triggered or Change in Data or ASDR Write Control, initiation of a transfer is controlled by the HT signal or by the detection of a change in data to be serialized or whenever a write operation to ASDR0 is detected.

42.5.3.6 Multiple Transfer Operation (MTO)

In DSI Configuration the MTO feature allows for multiple DSPIs within a device to be chained together in a parallel or serial configuration. The parallel chaining allows multiple DSPIs internal to a device and multiple SPI devices external to the device to

share SCK and PCS signals thereby saving the device pins. The serial chaining allows bits from multiple DSPIs to be concatenated into a single DSI frame. MTO is enabled by setting the MTOE bit in the DSPI_DSICR0.

In parallel and serial chaining there is one bus master and multiple bus slaves. The bus master initiates and controls the transfers, but the DSPI slaves generate trigger signals for the bus DSPI master when an internal condition in the slave warrants a transfer. The DSPI slaves also propagate triggers from other slaves to the master. When a DSPI slave detects a trigger signal on its HT input, the slave generates a trigger signal on the $\overline{\text{MTRIG}}$ output.

Serial and parallel chaining require multiplexing of signals external to the DSPI.

Note

TSB operation is not available in MTO mode. DSICR0[TSBC] and DSICR0[MTOE] bits should not be set simultaneously.

Note

DSI64 mode is not available in MTO mode. DSICR1[DSI64] and DSICR0[MTOE] bits must not be set simultaneously.

42.5.3.6.1 Parallel Chaining

Parallel chaining allows multiple DSPIs internal to a device and multiple SPI/DSI devices external to a device to share common SCK and PCS signals thereby saving pins. Two pins are saved per pair of DSPI/SPI. The following figure shows an example of how the blocks can be connected in a device.

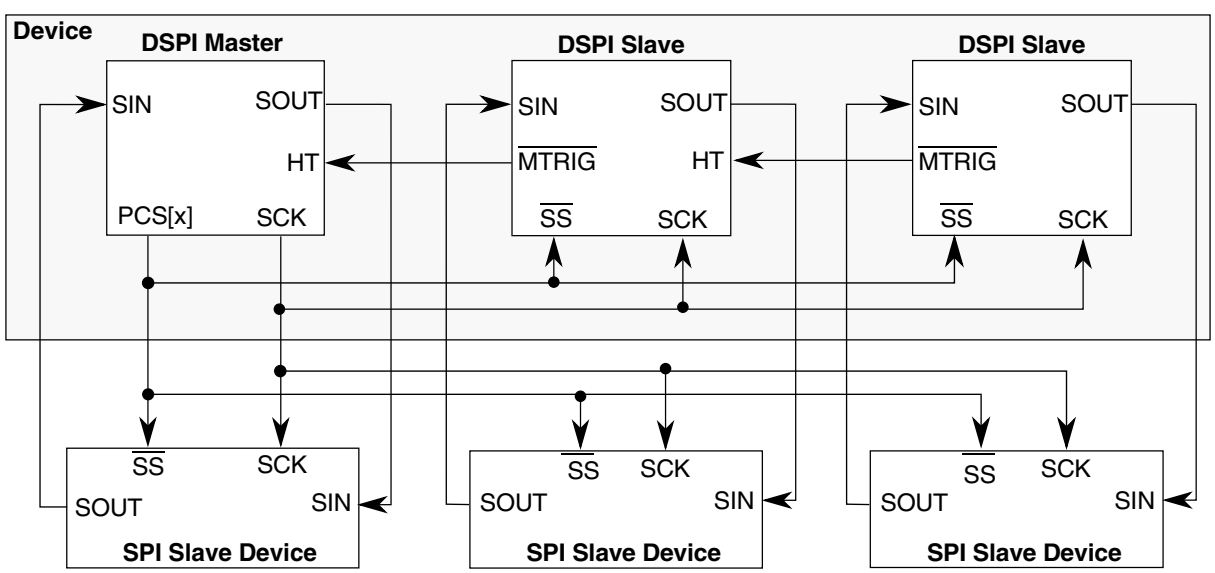


Figure 42-6. Example of Parallel Chaining of DSPIs

The DSPI master controls and initiates all transfer, but the DSPI slaves have a trigger output signal $\overline{\text{MTRIG}}$ that indicates to the master DSPI to start a transfer. When the DSPI slave has a change in its data to be serialized, it generates a pulse on the $\overline{\text{MTRIG}}$ signal to the master DSPI, which initiates the transfer. When a DSPI slave has its HT signal asserted it also generates a pulse on its $\overline{\text{MTRIG}}$ signal thereby propagating trigger signals from other DSPI slaves to the DSPI master.

To enable $\overline{\text{MTRIG}}$ signals generation the MTO has to be enabled for the DSPI slaves. The chained DSPI modules also should be in DSI configuration. The DSPI_DSICR0[MTOCNT] field must be written with the same number of bits to be transferred for each slave DSPI. The MTOCNT value must also match the FMSZ field of the DSPI_CTAR1 in the DSPI slaves and selected DSPI_CTAR register for the master DSPI module.

42.5.3.6.2 Serial Chaining

The serial chaining allows transfers of DSI frames of up to a total of 64 bits, using transfers of smaller DSI frames concatenated together by multiple DSPIs. The following figure shows an example of how the blocks can be connected in a device.

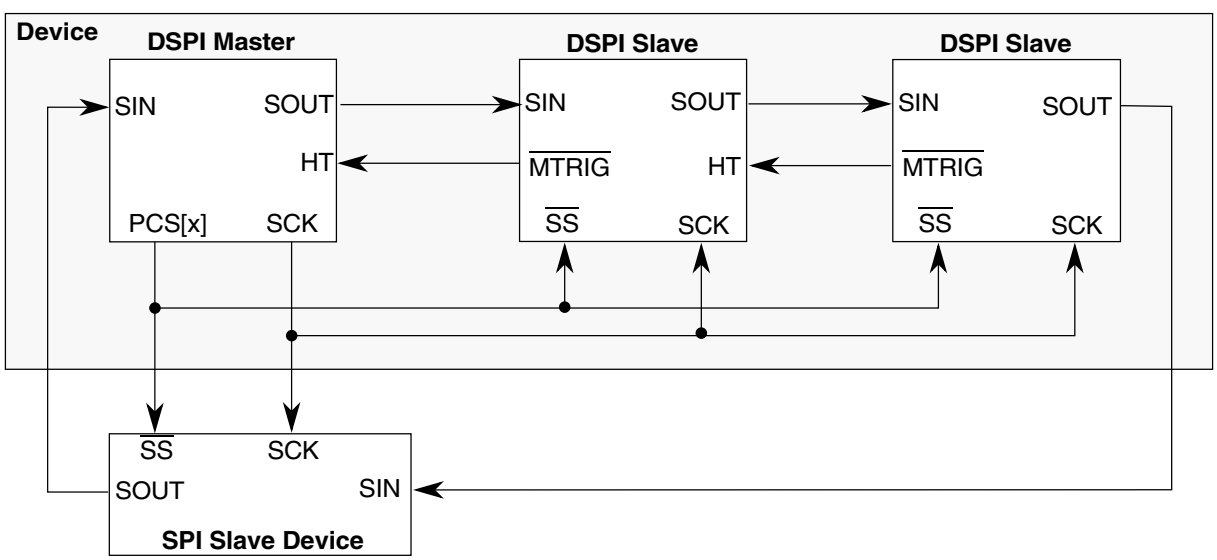


Figure 42-7. Example of Serial Chaining of DSPIs

The SOUT of the DSPI master is connected to the SIN of the first DSPI slave. The SOUT of the first DSPI slave is connected to the SIN input of the second slave and so on. The SOUT of the last DSPI slave is connected to the SIN of the external SPI slave. The SOUT of the external SPI slave is connected to the SIN of the DSPI master.

The DSPI master controls and initiates all transfers, but the slave DSPIs use the trigger output signal $\overline{\text{MTRIG}}$ to indicate to the DSPI master that a trigger condition has occurred. When a DSPI slave has a change in data to be serialized it asserts the $\overline{\text{MTRIG}}$ signal to

the DSPI master which initiates the transfer. When a DSPI slave has its HT signal asserted it will assert its $\overline{\text{MTRIG}}$ signal thereby propagating trigger signals from other DSPI slaves to the DSPI master.

The DSPI_DSICR0[MTOE] bit must be set in master and all slave devices. The DSPI_DSICR0[MTOCNT] field for the master and all slave devices must be written with the same value. The value must equal the sum of all FMSZ fields in the selected DSPI_CTAR registers for the DSPI master and all DSPI slaves. For example if one 16-bit DSI frame is created by concatenating eight bits from the DSPI master, and four bits from each of the DSPI slaves in the above figure, the DSPI master frame size must be set to eight in the FMSZ field, and the DSPI slaves frame size must be set to four. The largest DSI frame supported by the MTOCNT field is 64 bits. Any number of DSPIs can be connected together to concatenate DSI frames, as long as each DSPI transfers a minimum of 4 bits and a maximum of 32 bits and the total size of the concatenated frame is less than 65 bits long.

42.5.4 Combined Serial Interface (CSI) Configuration

The CSI Configuration of the DSPI is used to support SPI and DSI functions on a frame by frame basis. CSI Configuration allows interleaving of DSI data frames from the parallel input signals with SPI data frames from the TX FIFO. The data returned from the bus slave is either used to drive the Parallel Output signals or it is stored in the RX FIFO. The CSI Configuration allows serialized data and configuration or diagnostic data to be transferred to a slave device using only one serial link. The DSPI is in CSI Configuration when the DCONF field in the DSPI_MCR is 0b10. The following figure shows an example of how a DSPI can be used with a deserializing peripheral that supports SPI control for control and diagnostic frames.

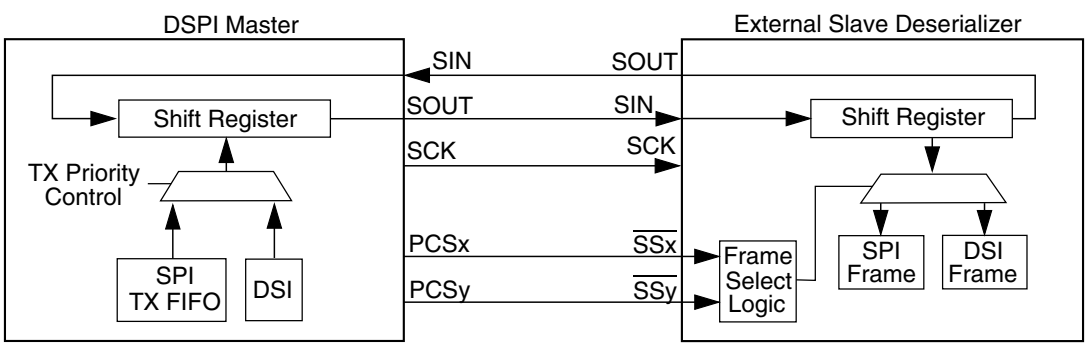


Figure 42-8. Example of System using DSPI in CSI Configuration

In CSI Configuration the DSPI transfers DSI data based on DSI Transfer Initiation Control .

When DSICR1[CSI_PRTY] is set to zero and there are SPI commands in the TX FIFO, the SPI frames have priority over the DSI frames. When the TX FIFO or CMD FIFO is empty, DSI transfer resumes.

When DSICR1[CSI_PRTY] bit is set, the DSI frames have priority over SPI frames. Thus, only when the DSI frame initiation requests have been catered to, will the SPI frames be transmitted, if available.

NOTE

The following combination is not a legal scenario and should be avoided - DSICR1[CSI_PRTY] = 1, DSICR1[TRRE] = 0 and DSICR1[CID] = 0, since such a configuration prevents SPI frames from ever getting transmitted.

Two peripheral chip select signals indicate whether DSI data or SPI data is transmitted. The user must configure the DSPI so that the two CTAR registers associated with DSI data and SPI data assert different peripheral chip select signals denoted in the figure as PCSx and PCSy. The CSI Configuration is only supported in master mode.

Data returned from the external slave while a DSI frame is transferred is placed on the Parallel Output signals. Data returned from the external slave while a SPI frame is transferred is moved to the RX FIFO. The TX FIFO, CMD FIFO and RX FIFO are fully functional in CSI mode.

42.5.4.1 CSI Serialization

Serialization in the CSI configuration is similar to serialization in DSI Configuration. The transfer attributes for SPI frames are determined by the DSPI_CTAR register selected by the CTAS field in the SPI command halfword. The transfer attributes for the DSI frames are determined by the DSPI_CTAR register selected by the DSICTAS field in the DSPI_DSICR0.

The Parallel Inputs signal states are latched into the DSPI DSI Serialization Data Register (DSPI_SDR0/1) and serialized based on the transfer initiation control settings in the DSPI_DSICR0.When SPI frames are written to the TX FIFO (and when CMD FIFO is not empty) they have priority over DSI data from the DSPI_SDR and are transferred at the next frame boundary.A copy of the most recently transferred DSI frame is stored in DSPI_COMPR0/1. The Transfer Priority Logic selects the source of the serialized data and asserts the appropriate PCS signal.

Separate frame completion interrupts are available to indicate frame transmission completion from SPI and DSI frames.

42.5.4.2 CSI Deserialization

The deserialized frames in CSI Configuration go into the DSPI_DDR0/1 or the RX FIFO based on the transfer priority logic.When DSI frames are transferred the returned frames are deserialized and latched into the DSPI_DDR0/1. When SPI frames are transferred the returned frames are deserialized and written to the RX FIFO.

42.5.5 Module baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the protocol clock frequency by a prescaler and a scaler with the option for doubling the baud rate. The following figure shows conceptually how the SCK signal is generated.

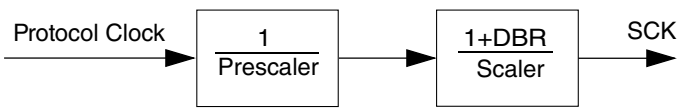


Figure 42-9. Communications clock prescalers and scalers

42.5.5.1 Baud rate generator

The baud rate is the frequency of the SCK. The protocol clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR, and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. The following table shows an example of how to compute the baud rate.

Table 42-9. Baud rate computation example

f_P	PBR	Prescaler	BR	Scaler	DBR	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

NOTE

The clock frequencies mentioned in the preceding table are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

42.5.5.2 PCS to SCK Delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See Figure 42-11 for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTAR*x* registers select the PCS to SCK delay by the formula in the CSSCK field description. The following table shows an example of how to compute the PCS to SCK delay.

Table 42-10. PCS to SCK delay computation example

f_{SYS}	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 µs

PCSCSK and CSSCK fields have no effect in TSB configuration.

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

42.5.5.3 After SCK Delay (t_{ASC})

The After SCK Delay is the length of time between the last edge of SCK and the negation of PCS. See Figure 42-11 and Figure 42-12 for illustrations of the After SCK delay. The PASC and ASC fields in the CTAR*x* registers select the After SCK Delay by the formula in the ASC field description. The following table shows an example of how to compute the After SCK delay.

Table 42-11. After SCK Delay computation example

f_P	PASC	Prescaler	ASC	Scaler	After SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 µs

PCASC and ASC fields have no effect in TSB configuration.

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

42.5.5.4 Delay after Transfer (t_{DT})

The Delay after Transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See Figure 42-11 for an illustration of the Delay after Transfer. The PDT and DT fields in the CTAR*x* registers select the Delay after Transfer by the formula in the DT field description. The following table shows an example of how to compute the Delay after Transfer.

Table 42-12. Delay after Transfer computation example

f_P	PDT	Prescaler	DT	Scaler	Delay after Transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

When in Non-Continuous Clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] field description. When in Continuous Clock mode and TSB is not enabled, the delay is fixed at 1 SCK period.

In TSB mode the Delay after Transfer is equal to a number formed by concatenation of PDT and DT fields plus 1 of the SCK clock periods. See detailed information on Timed Serial Bus (TSB)".

42.5.5.5 Peripheral Chip Select Strobe Enable ($\overline{\text{PCSS}}$)

The $\overline{\text{PCSS}}$ signal provides a delay to allow the PCS signals to settle after a transition occurs thereby avoiding glitches. When the Module is in Master mode and the PCSSE bit is set in the MCR, $\overline{\text{PCSS}}$ provides a signal for an external demultiplexer to decode peripheral chip selects other than PCS5 into glitch-free PCS signals. The following figure shows the timing of the $\overline{\text{PCSS}}$ signal relative to PCS signals.

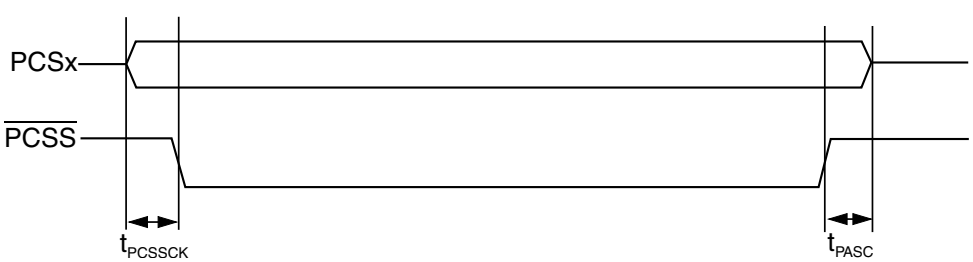


Figure 42-10. Peripheral Chip Select Strobe timing

The delay between the assertion of the PCS signals and the assertion of $\overline{\text{PCSS}}$ is selected by the PCSSCK field in the CTAR based on the following formula:

$$t_{\text{PCSSCK}} = \frac{1}{f_{\text{P}}} \times \text{PCSSCK}$$

At the end of the transfer, the delay between $\overline{\text{PCSS}}$ negation and PCS negation is selected by the PASC field in the CTAR based on the following formula:

$$t_{\text{PASC}} = \frac{1}{f_{\text{P}}} \times \text{PASC}$$

The following table shows an example of how to compute the t_{pcssck} delay.

Table 42-13. Peripheral Chip Select Strobe Assert computation example

f_P	PCSSCK	Prescaler	Delay before Transfer
100 MHz	0b11	7	70.0 ns

The following table shows an example of how to compute the t_{pasc} delay.

Table 42-14. Peripheral Chip Select Strobe Negate computation example

f_P	PASC	Prescaler	Delay after Transfer
100 MHz	0b11	7	70.0 ns

The $\overline{\text{PCSS}}$ signal is not supported when Continuous Serial Communication SCK or TSB mode is enabled.

NOTE

The clock frequency mentioned in the preceding tables is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

42.5.6 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In Master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTARn) select the polarity and phase of the serial clock, SCK.

- CPOL - Selects the idle state polarity of the SCK
- CPHA - Selects if the data on SOUT is valid before or on the first SCK edge

Even though the bus slave does not control the SCK signal, in Slave mode the values of CPOL and CPHA must be identical to the master device settings to ensure proper transmission. In SPI Slave mode, only CTAR0 is used. In DSI slave mode, only CTAR1 is used.

The module supports four different transfer formats:

- Classic SPI with CPHA=0
- Classic SPI with CPHA=1
- Modified Transfer Format with CPHA = 0
- Modified Transfer Format with CPHA = 1

A modified transfer format is supported to allow for high-speed communication with peripherals that require longer setup times. The module can sample the incoming data later than halfway through the cycle to give the peripheral more setup time. The MTFE bit in the MCR selects between Classic SPI Format and Modified Transfer Format.

In the interface configurations, the module provides the option of keeping the PCS signals asserted between frames. See Continuous Selection Format for details.

42.5.6.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in following figure is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

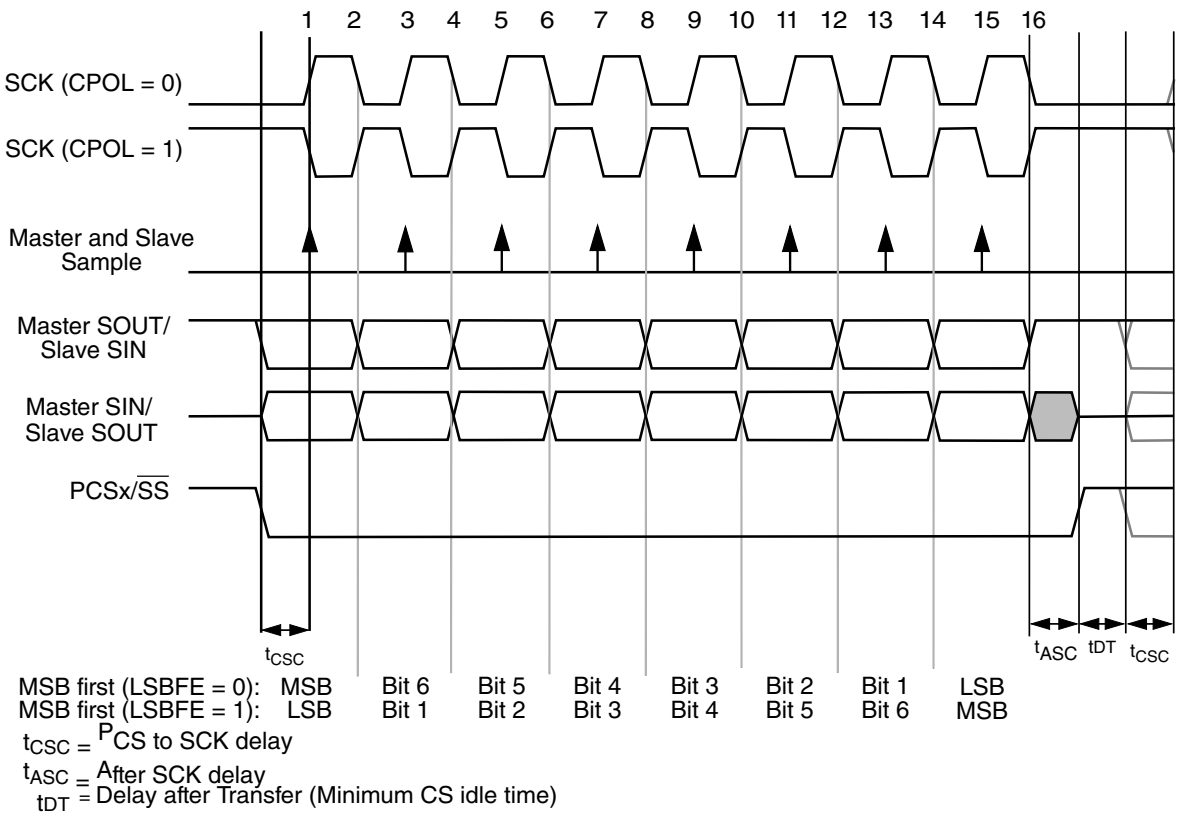


Figure 42-11. Module transfer timing diagram (MTFE=0, CPHA=0, FMSZ=8)

The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals. At the second edge of the SCK, the master and slave devices place their second data bit on their serial data output signals. For the rest of the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

42.5.6.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in the following figure is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format, the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges.

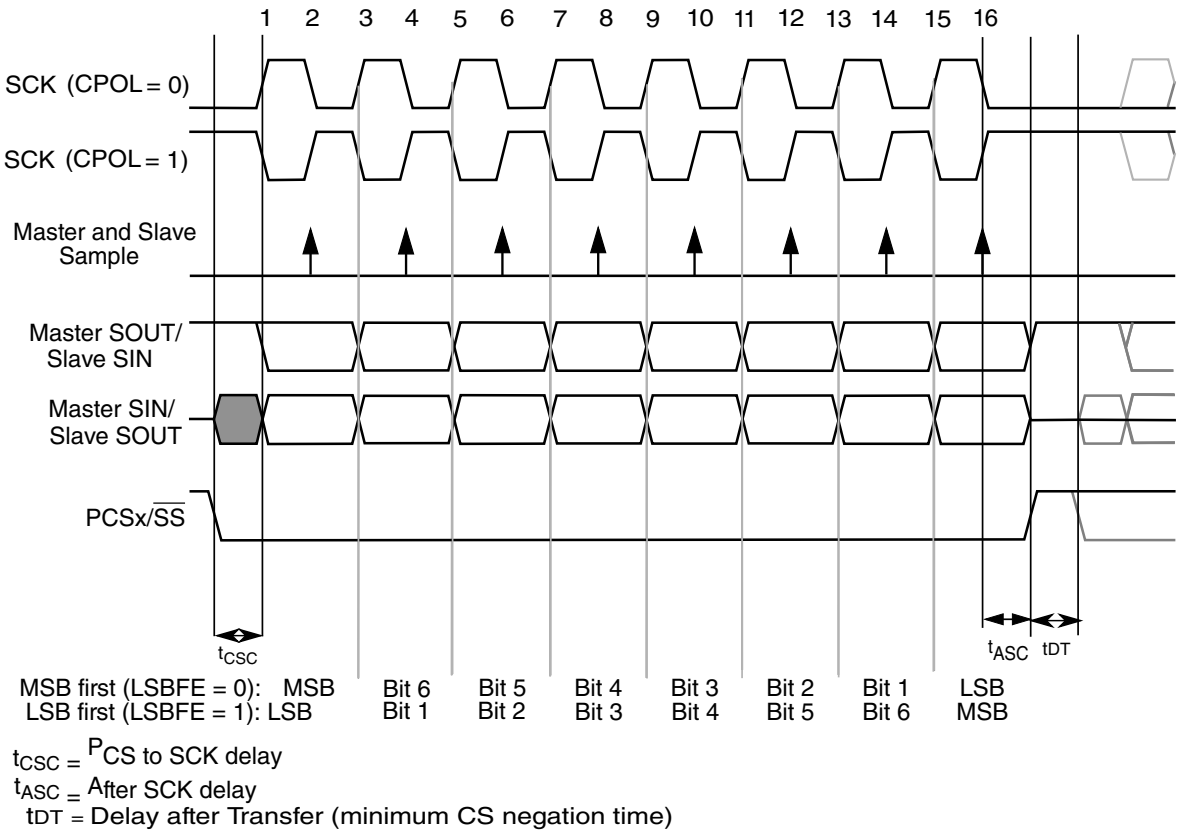


Figure 42-12. Module transfer timing diagram (MTFE=0, CPHA=1, FMSZ=8)

The master initiates the transfer by asserting the PCS signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins. For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

42.5.6.3 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 0)

In this Modified Transfer Format both the master and the slave sample later in the SCK period than in Classic SPI mode to allow the logic to tolerate more delays in device pads and board traces. These delays become a more significant fraction of the SCK period as the SCK period decreases with increasing baud rates.

The master and the slave place data on the SOUT pins at the assertion of the PCS signal. After the PCS to SCK delay has elapsed the first SCK edge is generated. The slave samples the master SOUT signal on every odd numbered SCK edge. The DSPI in the slave mode when the MTFE bit is set also places new data on the slave SOUT on every odd numbered clock edge. Regular external slave, configured with CPHA=0 format drives its SOUT output at every even numbered SCK clock edge.

The DSPI master places its second data bit on the SOUT line one protocol clock after odd numbered SCK edge if the protocol clock frequency to SCK frequency ratio is higher than three. If this ratio is below four the master changes SOUT at odd numbered SCK edge. The point where the master samples the SIN is selected by the DSPI_MCR[SMPL_PT] field. The master sample point can be delayed by one or two protocol clock cycles. The SMPL_PT field should be set to 0 if the protocol to SCK frequency ratio is less than 4. However if this ratio is less than 4, the actual sample point is delayed by one protocol clock cycle automatically by the design.

The following timing diagrams illustrate the DSPI operation with MTFE=1. Timing delays shown are:

- T_{csc} - PCS to SCK assertion delay
- T_{acs} - After SCK PCS negation delay
- T_{su_ms} - master SIN setup time
- T_{hd_ms} - master SIN hold time
- T_{vd_sl} - slave data output valid time, time between slave data output SCK driving edge and data becomes valid.
- T_{su_sl} - data setup time on slave data input
- T_{hd_sl} - data hold time on slave data input
- T_{sys} - protocol clock period.

The following figure shows the modified transfer format for CPHA = 0 and Fsys/Fsck = 4. Only the condition where CPOL = 0 is illustrated. Solid triangles show the data sampling clock edges. The two possible slave behavior are shown.

- Signal, marked "SOUT of Ext Slave", presents regular SPI slave serial output.
- Signal, marked "SOUT of DSPI Slave", presents DSPI in the slave mode with MTFE bit set.

Other MTFE = 1 diagrams show DSPI SIN input as being driven by a regular external SPI slave, configured according DSPI master CPHA programming.

Note

In the following diagrams, f_{sys} represents the protocol clock frequency from which the Baud frequency f_{sck} is derived.

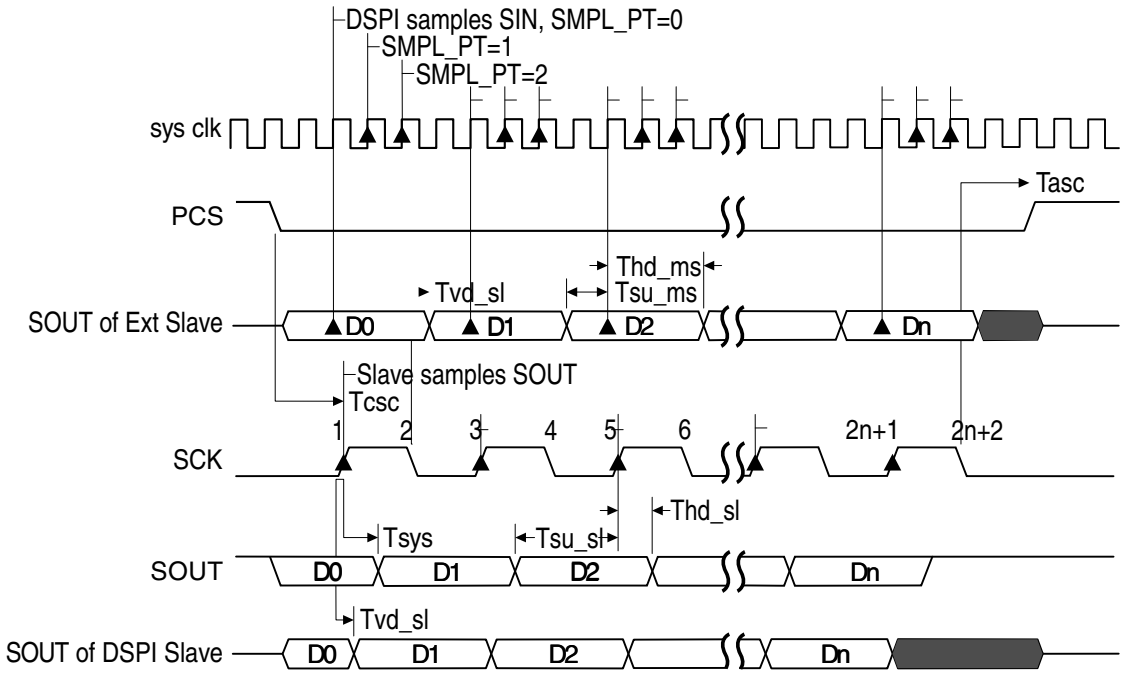


Figure 42-13. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/4$)

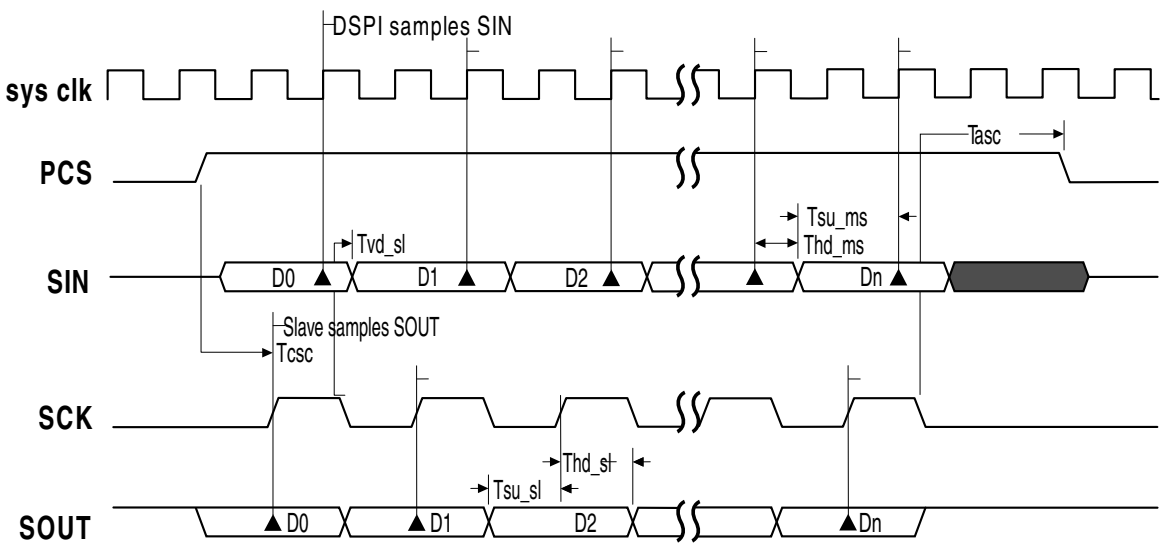


Figure 42-14. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/2$)

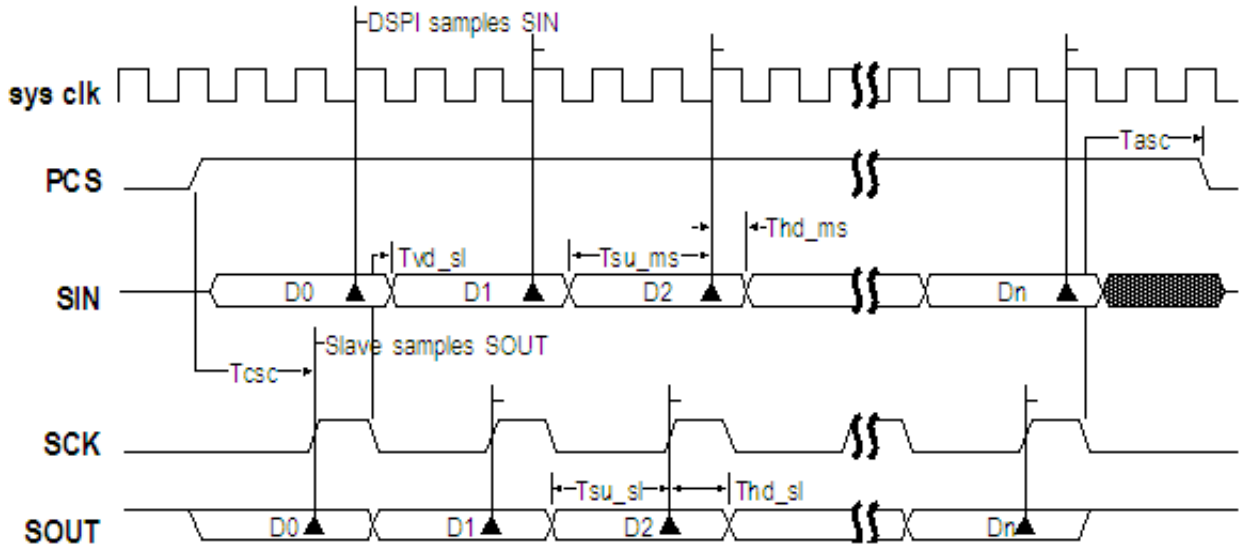


Figure 42-15. DSPI Modified Transfer Format (MTFE=1, CPHA=0, $f_{sck} = f_{sys}/3$)

42.5.6.4 Modified SPI/DSI Transfer Format (MTFE = 1, CPHA = 1)

The following figures show the Modified Transfer Format for CPHA = 1. Only the condition, where CPOL = 0 is shown. At the start of a transfer the DSPI asserts the PCS signal to the slave device. After the PCS to SCK delay has elapsed the master and the slave put data on their SOUT pins at the first edge of SCK . The slave samples the master SOUT signal on the even numbered edges of SCK. The master samples the slave SOUT signal on the odd numbered SCK edges starting with the third SCK edge. The slave samples the last bit on the last edge of the SCK. The master samples the last slave SOUT bit one half SCK cycle after the last edge of SCK. No clock edge will be visible on the master SCK pin during the sampling of the last bit. **The SCK to PCS delay and the After SCK delay must be greater or equal to half of the SCK period.**

NOTE

When MTFE=1 with continuous SCK enabled (MCR [CONT_SCKE] =1) in master mode, configure CTAR[LSBFE]=0 for correct operations while receiving unequal length frames. If PUSHR[CONT] is also set for back to back frame transfer, also configure the frame size of the first frame as less than or equal to the frame size of the next frame. In this scenario, make sure that for all received frames, the bits are read equal to their respective frame sizes and any extra bits during POP operation are masked.

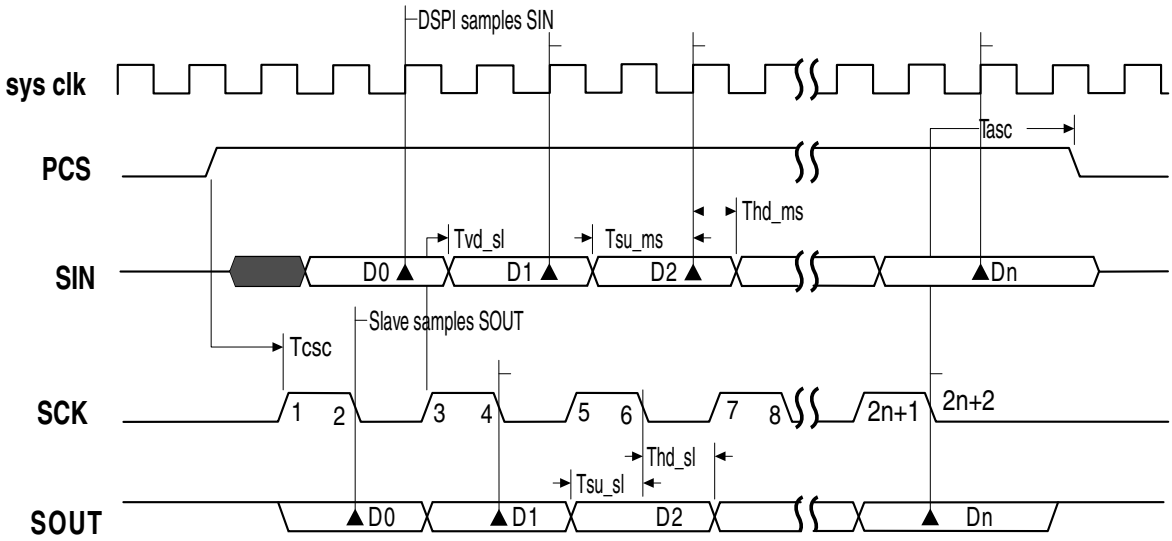


Figure 42-16. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/2$)

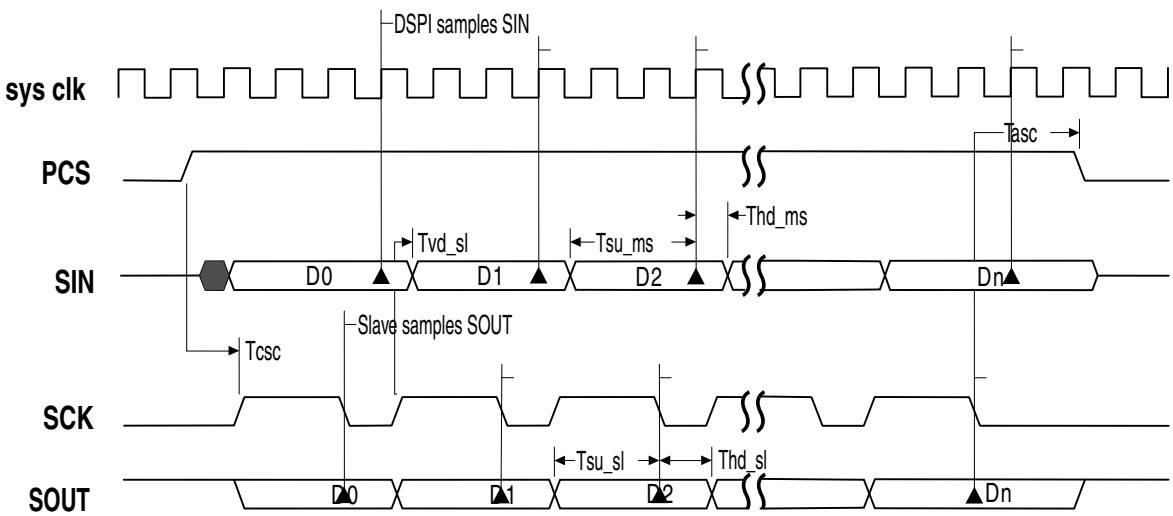


Figure 42-17. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/3$)

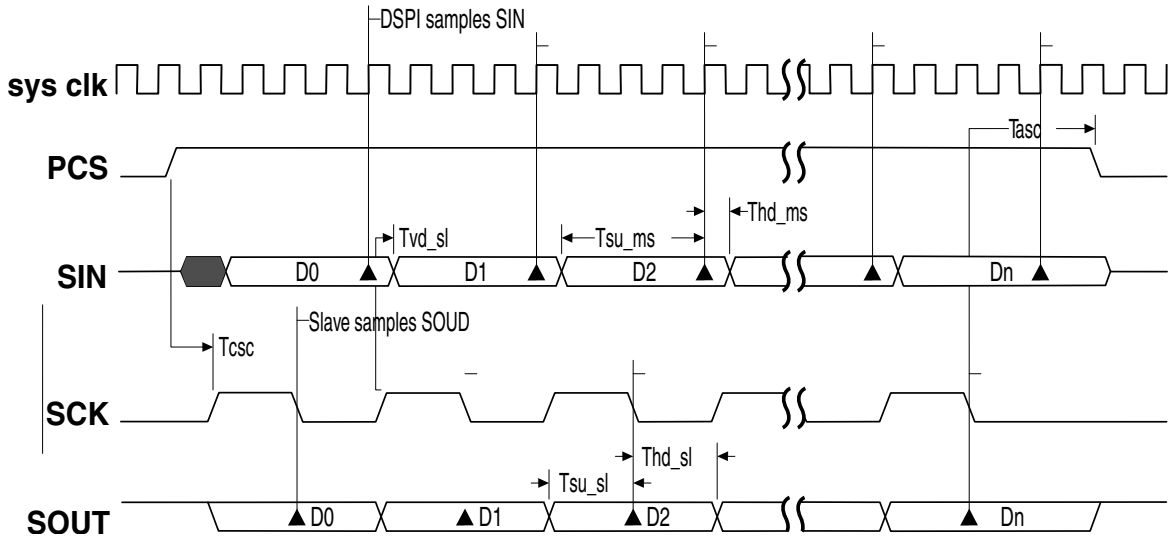


Figure 42-18. DSPI Modified Transfer Format (MTFE=1, CPHA=1, $f_{sck} = f_{sys}/4$)

42.5.6.5 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle both the following cases. The Continuous Selection Format is enabled for the SPI configuration by setting the CONT bit in the SPI command. Continuous Selection is enabled for the DSI Configuration by setting the DCONT bit in the DSICR0.The behavior of the PCS signals in the two configurations is identical so only SPI configuration will be described.

When the CONT bit = 0, the module drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSISn bits in the MCR. The following timing diagram is for two four-bit transfers with CPHA = 1 and CONT = 0.

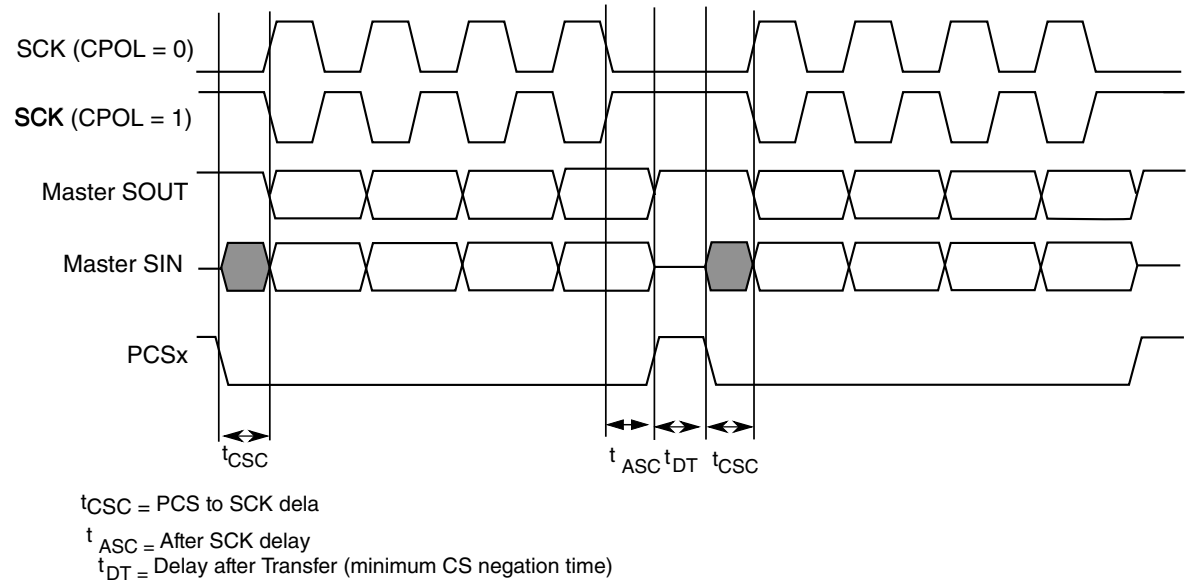


Figure 42-19. Example of non-continuous format (CPHA=1, CONT=0)

When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers (t_{DT}) is not inserted between the transfers. The following figure shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

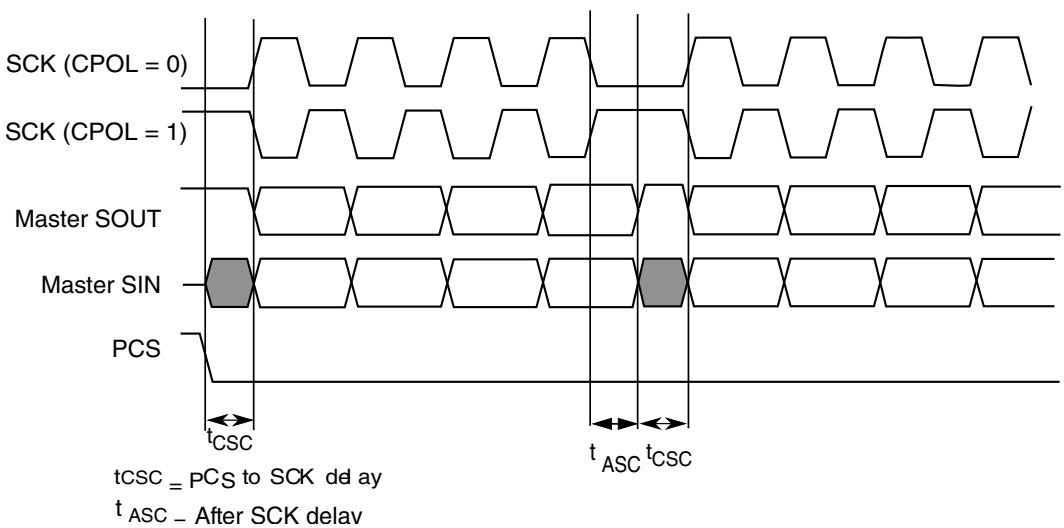


Figure 42-20. Example of continuous transfer (CPHA=1, CONT=1)

When using the module with continuous selection follow these rules:

- All transmit commands must have the same PCSn bits programming.
- The CTARs, selected by transmit commands, must be programmed with the same transfer attributes. Only FMSZ field can be programmed differently in these CTARs.

- When transmitting multiple frames in this mode, the user software must ensure that the last frame has the PUSHR[CONT] bit deasserted in Master mode and the user software must provide sufficient frames in the TX_FIFO to be sent out in Slave mode and the master deasserts the PCSn at end of transmission of the last frame.
- PUSHR[CONT] and DSICR0[DCONT] must be deasserted before asserting MCR[HALT] in master mode. This will make sure that the PCSn signals are deasserted. Asserting MCR[HALT] during continuous transfer will cause the PCSn signals to remain asserted and hence Slave Device cannot transition from Running to Stopped state.

NOTE

User must fill the TX FIFO with the number of entries that will be concatenated together under one PCS assertion for both master and slave before the TX FIFO becomes empty.

When operating in Slave mode, ensure that when the last entry in the TX FIFO is completely transmitted, that is, the corresponding TCF flag is asserted and TXFIFO is empty, the slave is deselected for any further serial communication; otherwise, an underflow error occurs.

42.5.6.6 Fast Continuous Selection Format

The Fast Continuous Selection Format functions similar to Continuous Selection Format except that the inter command delays, t_{ASC} and t_{CSC}, can be masked out and are not inserted by the hardware.

NOTE

The Fast Continuous Selection Format is available in the SPI configuration only and when Continuous Serial Communication Clock mode is disabled. Masking of delays is not allowed in DSI and CSI configurations or if the transfer is non-continuous.

The Fast Continuous Selection Format is enabled by writing ‘1’ into FCPCS bit of the MCR register. When this bit is asserted, MASC and MCSC bits of the PUSHR register perform the function of mask bits for the transmit frame. These bits individually mask the t_{ASC} and t_{CSC} delays as programmed by the user software. A normal Continuous Selection Format has these two delays for each frame that is transmitted with the CONT

bit asserted. In order to avoid these delays and to speed up the transfer process, the software can simply mask these delays while programming the command in the PUSHR register.

While masking the delays, the software must follow the following masking rules, else correct operation is not guaranteed.

- MASC bit masks the “After SCK” delay for the current frame.
- MCSC bit masks the “PCS to SCK” delay for the next frame.
- “After SCK” (t_{ASC}) delay must not be masked when the current frame is the last frame in the continuous selection format.
- The “PCS to SCK” delay for the first frame in the continuous selection format cannot be masked.
- Masking of only t_{ASC} is not allowed. If t_{ASC} is masked then t_{CSC} must be masked too.
- Masking of both t_{ASC} and t_{CSC} delays is allowed. In this case, the delay between two frames is equal to half the baud rate set by the user software.
- Masking of only t_{CSC} is allowed. In this case, the delay between two frames is equal to the t_{ASC} time and thus the user software must ensure that the t_{ASC} time is greater than the baud rate.
- The user software must not mask these delays if the continuous selection format is not used and MCR[FCPCS] is asserted.
- Rules applicable to the Continuous Selection Format are applicable here too.

The following figure shows the timing for a Fast Continuous Selection Format transfer. Here seven frames are transferred with both t_{ASC} and t_{CSC} delays masked except for the last frame that terminated the transfer. The last frame has t_{ASC} delay at its end.

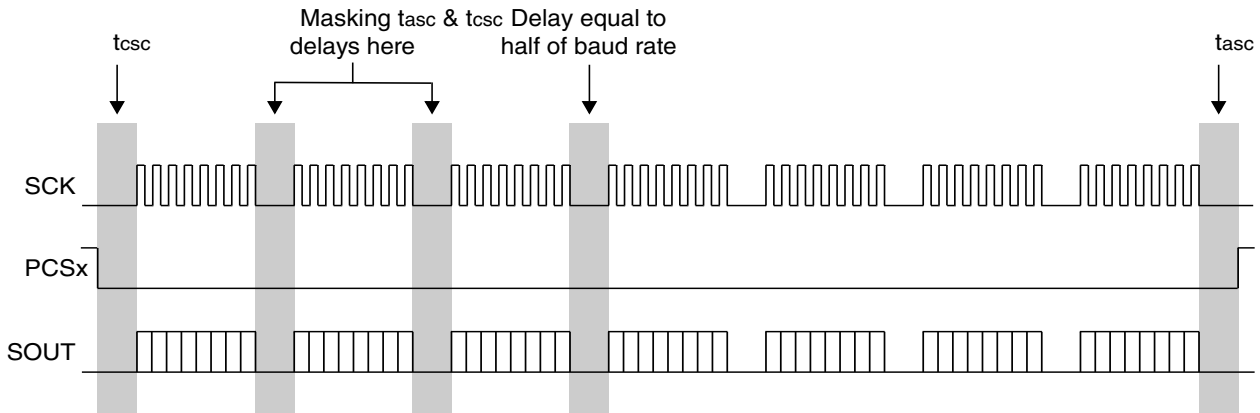


Figure 42-21. Example of Fast Continuous Selection Format

In case any chip select is to be changed, then the fast continuous selection format should be terminated and then the chips selects should change and appropriate delays must be introduced.

42.5.7 Continuous Serial Communications Clock

The module provides the option of generating a Continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous SCK only if MCR[HALT] bit is low. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA=1. Clearing CPHA is ignored if the CONT_SCKE bit is set. Continuous SCK is supported for Modified Transfer Format.

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the module is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- When the module is in DSI configuration, the CTAR specified by the DSICTAS field is used at all times.
- When the module is in CSI configuration, the CTAR selected by the DSICTAS field is used initially.

- At the start of a SPI frame transfer, the CTAR specified by the CTAS value for the frame is used.
- At the start of a DSI frame transfer, the CTAR specified by the DSICTAS field is used.

- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the module is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle. When TSB configuration is enabled the t_{DT} is programmable from 1 to 64 SCK cycles. The following figure is the timing diagram for Continuous SCK format with Continuous Selection disabled.

NOTE

In Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TX FIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

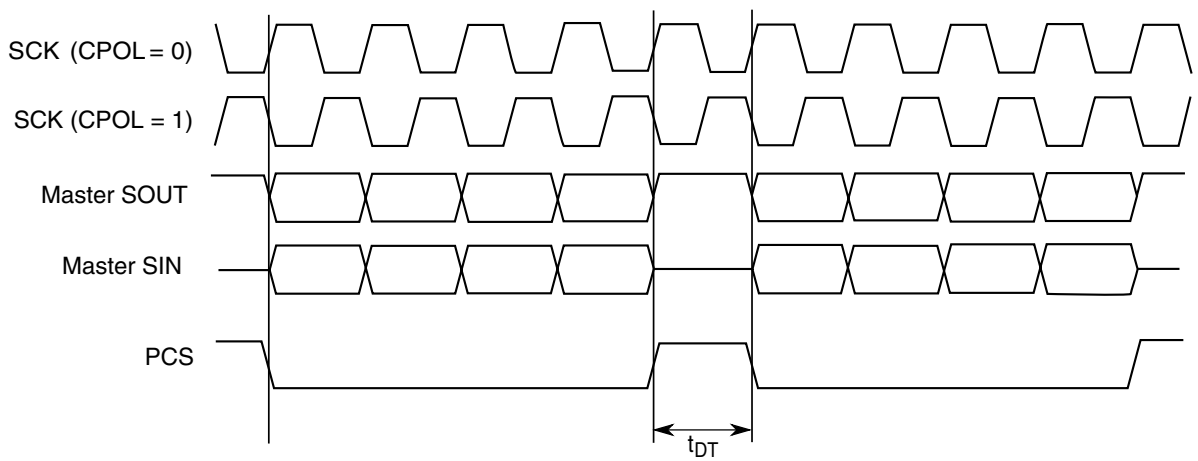


Figure 42-22. Continuous SCK Timing Diagram (CONT=0)

If the CONT bit in the TX FIFO entry is set or the DCONT in DSICR0 is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT, that is, SOUT pulled high. This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the TX FIFO.

- Continuous SCK with CONT bit set and entering Stopped state (refer to Start and Stop of module transfers).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

The following figure shows timing diagram for Continuous SCK format with Continuous Selection enabled.

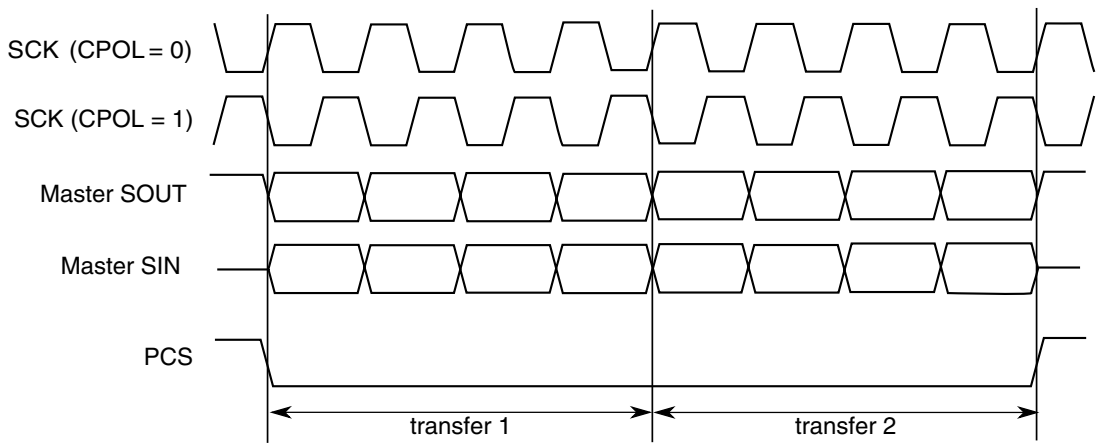


Figure 42-23. Continuous SCK timing diagram (CONT=1)

42.5.8 Slave Mode Operation Constraints

Slave mode logic shift register is buffered. This allows data streaming operation, when the module is permanently selected and data is shifted in with a constant rate.

The transmit data is transferred at second SCK clock edge of the each frame to the shift register if the $\overline{SS}$ signal is asserted and any time when transmit data is ready and $\overline{SS}$ signal is negated.

Received data is transferred to the receive buffer at last SCK edge of each frame, defined by frame size programmed to the CTAR0/1 register. Then the data from the buffer is transferred to the RXFIFO or DDR register.

If the $\overline{SS}$ negates before that last SCK edge, the data from shift register is lost.

42.5.9 Timed Serial Bus (TSB)

The DSPI can be programmed in Timed Serial Bus configuration by setting the TSBC bit in the DSPI_DSICR0 register. See DSPI DSI Configuration Register 0 (DSPI_DSICR0) for details.

TSB configuration provides the Micro Second Channel (MSC) downstream channel support.

The MSC upstream channel is not supported by the DSPI, but can be supported by any available Serial Communication Controller (SCI or UART) in the SoC.

The DSI Continuous Frame Selection Format is not supported by the TSB Mode (DSICR0[DCONT] bit cannot not be asserted).

To work in TSB mode the DSPI must be in master mode and in DSI (DCONF = 0b01) or CSI (DCONF = 0b10) configuration. Both Continuous and Non Continuous Serial Communication Clock (controlled by the DSPI_MCR[CONT_SCKE] bit) are supported in the TSB mode.

The following figure shows the signals used in the TSB interface.

In the TSB configuration the DSPI is able to send from 4 to 66 bits MSC data frames (4 to 64 serialized data bits and up to 2 Data Selection zero bits). The serialized data bits source can be either:

- the DSPI DSI Alternate Serialization Data Register (DSPI_ASDR0/1), written by the host software,
- Parallel Input pin states latched into the DSPI DSI Serialization Data Register (DSPI_SDR0/1).

DSPI_DSICR0 TXSS bit or DSPI_SSR0–1 register bits define the source of the data.

The Least Significant Bits of the DSPI_ASDR0 or DSPI_SDR0 registers are selected to be serialized if the data frame is set to less than 32 bits.

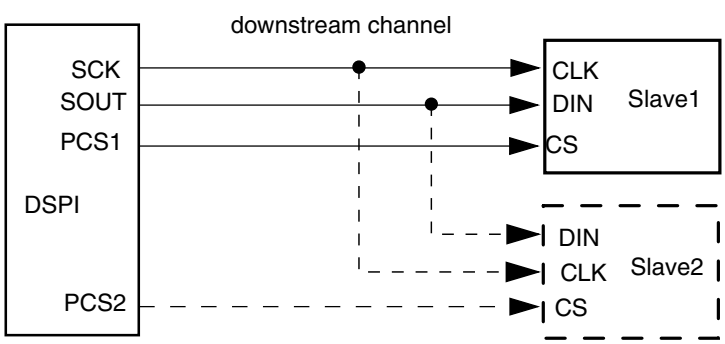


Figure 42-24. DSPI usage in the TSB Configuration

The PCS signals are driven together with SOUT. The t_{CSC} and t_{ASC} delays are not available. Delay after Transfer (DT) is set in SCK clock periods as a binary number formed by concatenation of the DSPI_CTARn PDT and DT fields plus one, allowing to set DT from 1 to 64 serial clock periods. DT field provides least significant bits and PDT field provides most significant bits of the Delay after Transfer.

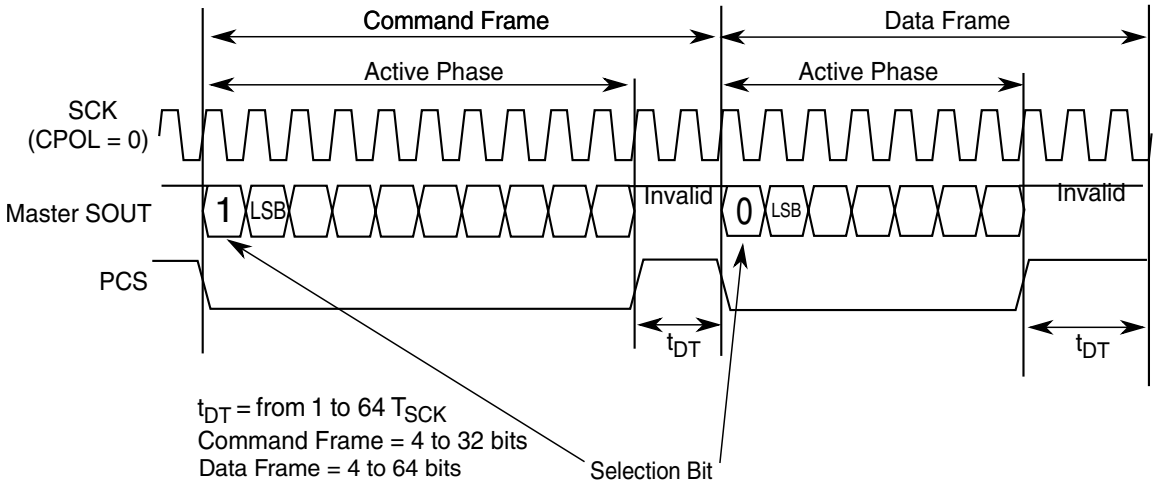


Figure 42-25. TSB Downstream Frames

Above figure shows the two types of MSC downstream frames - command frame, and data frame.

The first transmitted bit, called the selection bit, determines the frame type:

- The selection bit "0" indicates a data frame
- The selection bit "1" indicates a command frame

Data frame may contain up to 2 selection bits to support two external slave devices, (so called dual receiver configuration) or no selection bits at all.

The command frame can be written by software, through SPI TX FIFO, using one or two FIFO entries with help of the CONT bit or MCR[XSPI]. The data frame consists of up to 64 bits from the SDR0–1 or ASDR0–1 registers and up to two selection bits (0). Number of data bits in the data frame is defined by the DSCICR1[TSBCNT] field.

The selection bit of the MSC command frames (1) can be implemented by software.

The selection bits in the data frames are enabled by DSPI_DSICR1 DSE0 and DSE1 bits. Each DSEn bit set increases the data frame size by one bit.

To comply with MSC specification, set DSPI_CTARn[LSBFE] to transmit least significant bit first.

Regardless the LSBFE bit setting, the Data Frame Selection Bits, if enabled, are always transmitted first, before corresponded data sub frames.

42.5.9.1 MSC Dual Receiver Support with PCS Switch Over

When in TSB mode it is possible to switch the set of PCS signals that are driven during the first part of the frame to a different set of PCS signals during the second part of the frame. The bit, at which this switch over occurs, is defined by FMSZ field of the DSPI_CTARn register, selected by DSICTAS field of the DSICR0 register. If DSICR1[DSI64E] is enabled, then the bit at which the PCS switch occurs, is defined by the concatenation of {DSICR0[FMSZ4], CTAR[FMSZ]} fields.

Number of the bits, not including the Data Selection Bit, in the first part of the frame is equal to value of the FMSZ field plus one. (This value changes to {FMSZ4,FMSZ} plus one, when DSICR1[DSI64E] is enabled). During this part of the frame the PCS signal levels are controlled by DSPI_DSICR0 DPCSn bits, after that by DSPI_DSICR1 DPCS1_n bits.

The PCS switch over occurs at driving edge of the SCK clock output.

The second Data Selection Bit is inserted after the PCS switch over if enabled.

Data Frame with PCS switch over is shown in the following figure.

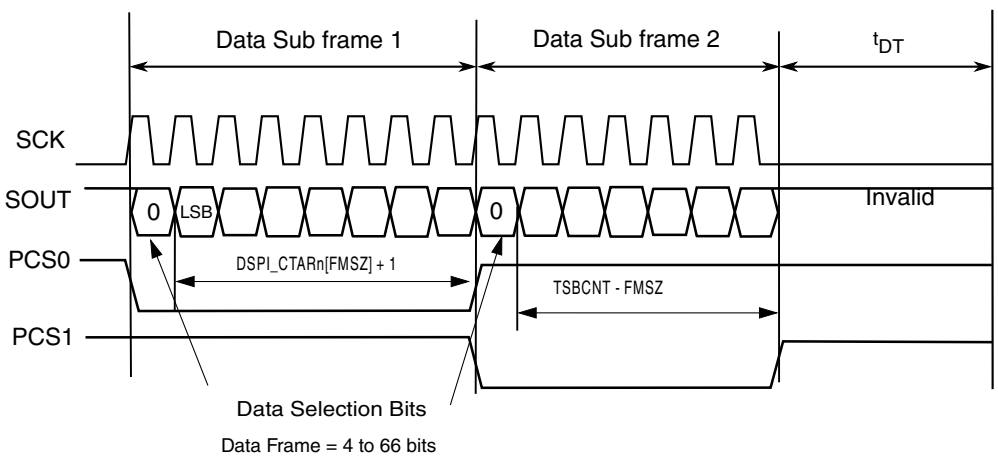


Figure 42-26. TSB Data Frame Format for MSC Dual Receiver Operation

42.5.10 Interleaved TSB (ITSB) Mode

This mode is similar to the TSB mode in some aspects except for the differences mentioned in the table below. The major difference is that the ITSB mode is highly generic in its approach to be compliant with the Downstream operation of the

Microsecond Channel Protocol. It includes a triggered time-slot based system with user configurable options for the frames to be transmitted in each time slot. (The time period between any two triggers is known as a time-slot).

The DSPI can be programmed to the ITSB Mode by asserting DSICR0[TSBC], DSICR0[ITSB]. To work in ITSB Mode, the DSPI must be in Master Mode and MCR[DCONF] must be set to CSI mode. (DCONF = 0b10). Both continuous and non-continuous clock serial communication clock (MCR[CONT_SCKE] are supported in this mode.

There are three types of frames which can be transmitted in a time slot i.e. Data Frame, Command Frame and Passive (Idle) frame. Similar to the TSB Mode, SPI frames are used for MSC Command frames, while DSI frames are used for MSC Data frames. A Passive Frame is an Idle frame in which no chip selects are asserted and no data is transmitted. The frame transmission is broadly governed by the following rules:

- Transmission of frames is done on trigger pulses that can be generated internal to the DSPI. The internal trigger is periodic with a period of 4. Thus there are 4 time-slots which appear periodically in the ITSB mode. The frame rules to be followed during each of the 4 time-slots can be programmed via TS_CONF[TSn] fields. The length of each time slot can be programmed via TSL[TSx_LEN] fields. The minimum length of a time slot is governed by the following formulae :
  - Minimum Length for Time slot X = Frame size of the frame to be transmitted in Time slot X + no. of selection bits, if any + 2 (Inherent time(Delay) changed by baud rate(Fp/n)).
  - The above constraint is only true for Data and Command frames. Passive frames do not have a minimum trigger length constraint. The user software shall configure the trigger lengths and trigger source.
- If the previously transmitted frame was a Command (SPI) frame or the TX_FIFO (or CMD FIFO) is empty, the non-passive frame transmitted on a subsequent trigger will not be a command frame.
- If the previously transmitted frame was not a Command (SPI) frame and the TX_FIFO (and CMD FIFO) have sufficient data to allow a SPI frame transmission, the frame transmitted on subsequent trigger can be a Command (SPI) frame.
- A Dual Receiver Frame (DRF) is a single Data (DSI) Frame which is split and transmitted to two different receivers (via occurrence of PCS switchover at the split) - the former part being called DRF1 and the latter part as DRF2 (Dual Receiver Frames 1 & 2). Using TS_CONF[TSn] fields, a time-slot can be programmed to transmit a sole DRF1 or a DRF2 frame. A sole DRF1 frame followed by any other frame is a valid scenario. A DRF2 frame programmed for a time slot is only valid if

the immediate previous time-slot was DRF1 - an invalid DRF2 frame, if programmed results in the transmission of a Passive frame (PF).

This mode does not change the way DSI and SPI frame transmissions are done except that ITSB mode interleaves the source (SPI or DSI) with a configurable set of rules based on triggered time-slot system. All features of SPI and DSI mode can be used except for the Continuous Frame Selection format (PUSHR[CONT] and DSICR0[DCONT] bits should not be asserted).

MCR[MTFE] should be set to 0 during ITSB Mode of operation, since MicroSecond Channel upstream operation is not supported.

The following table lists the differences between the TSB and ITSB modes:

Table 42-15. Differences between TSB and ITSB modes

TSB Mode	ITSB Mode
Priority available to frames from SPI. Back to back SPI frames are sent out.	No priority. Messages are interleaved such that there are no back-to-back SPI frames
Selection Bit inserted for DSI frames only. Selection bit for SPI frames is inserted by software.	Selection bit for both DSI and SPI frames inserted automatically. Encoding of selection bit same as TSB mode (0 = DSI; 1 = SPI).
Occasionally triggered transmission (that is trigger is received via hardware trigger input).	Triggered time-slot based transmission (that is frames are transmitted continuously on every trigger).
Gap between frames is configurable. Time between frames is t_{DT} which is configurable from 1 to 64 baud rate clock cycles.	Gap between frames is configurable and can be introduced in any of the two ways - By programming a Passive Frame or by increasing the length of a time-slot. Time t_{DT} is not used in this mode as it can be made part of the trigger period (as passive time).

In both TSB & ITSB mode, separate frame completion interrupts will be available to indicate frame transmission completion from SPI & DSI frames. MSC dual receiver support with PCS switchover is also supported in ITSB mode. See MSC Dual Receiver Support with PCS Switch Over for more details.

The ITSB mode is suitable to implement the Downstream Channel for the MSC Bus because of the above features.

42.5.10.1 Configuring DSPI for ITSB mode

The following steps give the recommended way to initialize the DSPI for the ITSB mode of operation:

- The DSPI should be put in HALT mode (by asserting the MCR[HALT] bit) before programming the registers.

- Set the DSICR0[TSBC] and DSICR0[ITSB] bits. The DSICR0[TRRE] and DSICR0[CID] bits should be cleared. DSICR0[DCONT] and PUSHR[CONT] should not be used in ITSB mode. The bit MCR[MTFE] should be cleared as Upstream operation is not supported in ITSB Mode. If DSICR0[ITSB] bit is set without setting DSICR0[TSBC], then the DSPI operates in the Normal Mode depending on MCR[DCONF] bit.
- Configure the Time Slot Length's via TSL register and the frame rules for each time slot via TS_CONF register.
- Based on trigger source requirements, set the values in the DSICR0[TRG] bit.
- Enable frame completion interrupts for either any frame completion (TCF) or separate frame complete (DSITCF and SPITCF) as required by application.
- When remainder configuration of DSPI is complete, clear the MCR[HALT] bit to allow DSPI to start operations.

Once configured and DSPI is brought out of HALT, the ITSB mode shall start sequencing the frames from either DSI or SPI as per the rules mentioned in Interleaved TSB (ITSB) Mode.

42.5.10.2 Variable Length Time Slots

In the Interleaved TSB Mode, the TSL[TSx_LEN] registers are used to configure the time slot length's as required by the application. There are up to 4 time slots named TS0, TS1, TS2 and TS3 respectively, which repeat periodically. To achieve a periodicity less than 4, any time slot length can be programmed as 0 (which eliminates this time slot). The following figures illustrate this concept.

In the figure below, the fields TSL[TSx_LEN] are programmed to be non-zero for x = 0,1,2,3. Hence, the time slots repeat after every 4 trigger pulses as shown.

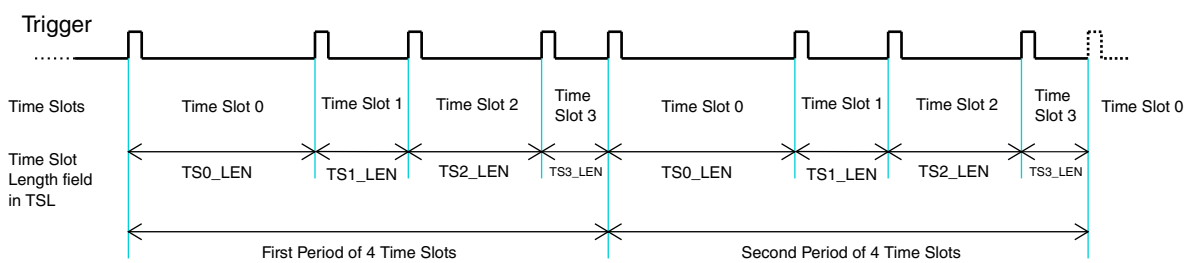


Figure 42-27. Periodic Time Slots with period = 4

In the figure below, the fields TSL[TSx_LEN] are programmed to be non-zero for x = 0,1,2, while TSL[TS3_LEN] = 0. Hence, the time slots repeat after every 3 trigger pulses as shown.

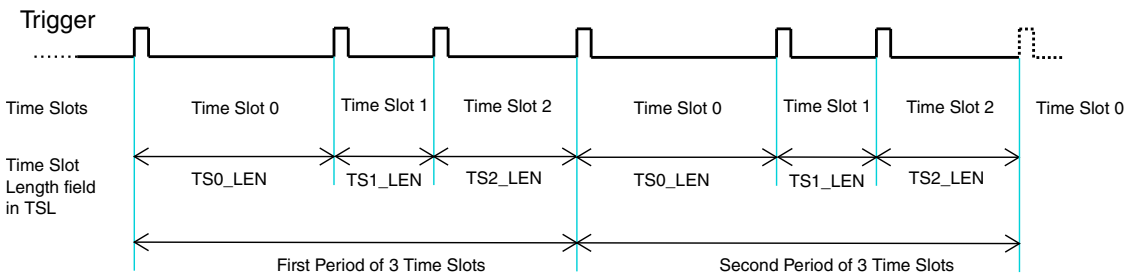


Figure 42-28. Periodic Time Slots with period = 3

The following figure shows time slots with periodicity of 1. Here the fields TSL[TSx_LEN] are programmed to zero for x = 1,2 and 3 while TSL[TS0_LEN] is non-zero. Hence, the time slots repeat after every trigger as shown.

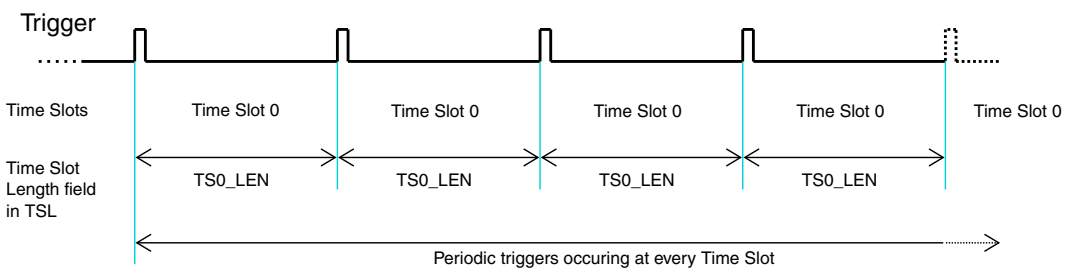


Figure 42-29. Periodic Time Slots with period = 1

Similarly, any TSL[TSx_LEN] can be made 0 to vary periodicity of time slots, but it is an illegal scenario to have all TSL[TSx_LEN] fields as 0.

NOTE

If external triggering mechanism is enabled (DSICR[TRG] is set), then it must be ensured that the minimum distance between two trigger pulses is at least 5P, where P is the protocol clock period.

When internal triggering mechanism is enabled (DSICR[TRG] is zero), the minimum value which can be programmed in the field TSL[TSx_LEN] is given by L = F + 1 + m; where F is the frame size of the frame programmed to be transmitted for the time slot TSx. The variable 'm' equals 1 when maximum baud rate is programmed (i.e. SCK = protocol clock frequency/ 2) and 0 otherwise.

42.5.10.3 Using ITSB mode for MSC Downstream Channel

MSC Downstream Channel uses three types of frames: command frames, data frames and passive frames. The Downstream operation of these frames require:

- Data frames can be sent back to back.
- Back to back command frames are not a legal scenario.
- Passive frames are used when no activity is desired for any time slot. In a passive frame, chip selects are de-asserted, and no SCK or data is transmitted. For this reason Command Frames cannot be separated by Passive frames as it would still be an illegal scenario.
- In dual MSC ASIC, some bits of the data frame goes to first ASIC and remainder bits to second ASIC. The first part is referred to as DRF1 (Dual Receiver Frame 1) and the second part as DRF2 (Dual Receiver Frame 2).
- A DRF2 programmed for a time slot is only valid if the immediately preceding time slot was DRF1. If DRF2 is valid, it would be transmitted seamlessly after DRF1 without waiting for the trigger pulse for the time slot of DRF2. For that reason, the TSL[TSx_LEN] for DRF1 must be programmed to the minimum possible value. If DRF2 is not valid, a Passive Frame is transmitted instead.

The ITSB mode can help implement the MSC Downstream Channel by allowing the command frames to be sent via the SPI interface and data frames to be sent via the DSI interface. Once the DSPI has been configured to work in ITSB mode (See Configuring DSPI for ITSB mode), the user software can program the DSI and SPI sides to send data and command frames respectively. The programming of frames for transmission via DSI or SPI is same as explained in Combined Serial Interface (CSI) Configuration.

The following figure shows a sample MSC Downstream transmission for the ITSB mode with the following configuration:

- All TSL[LEN] are programmed to be non-zero. Note that the length between triggers are not drawn to scale in the figure. The lengths of all 4 time slots shown are unequal.
- Data and Command selection bits are enabled, i.e. DSICR1[DSE0] and DSICR1[CSE] are set.
- TS_CONF[TS0] = CMD_SRDF
- TS_CONF[TS1] = CMD_SRDF
- TS_CONF[TS2] = CMD_PF

- TS_CONF[TS3] = CMD_SRDF
- PCS0 is used for Data Frames while PCS1 is used for Command Frames. Both chip select's are active low.

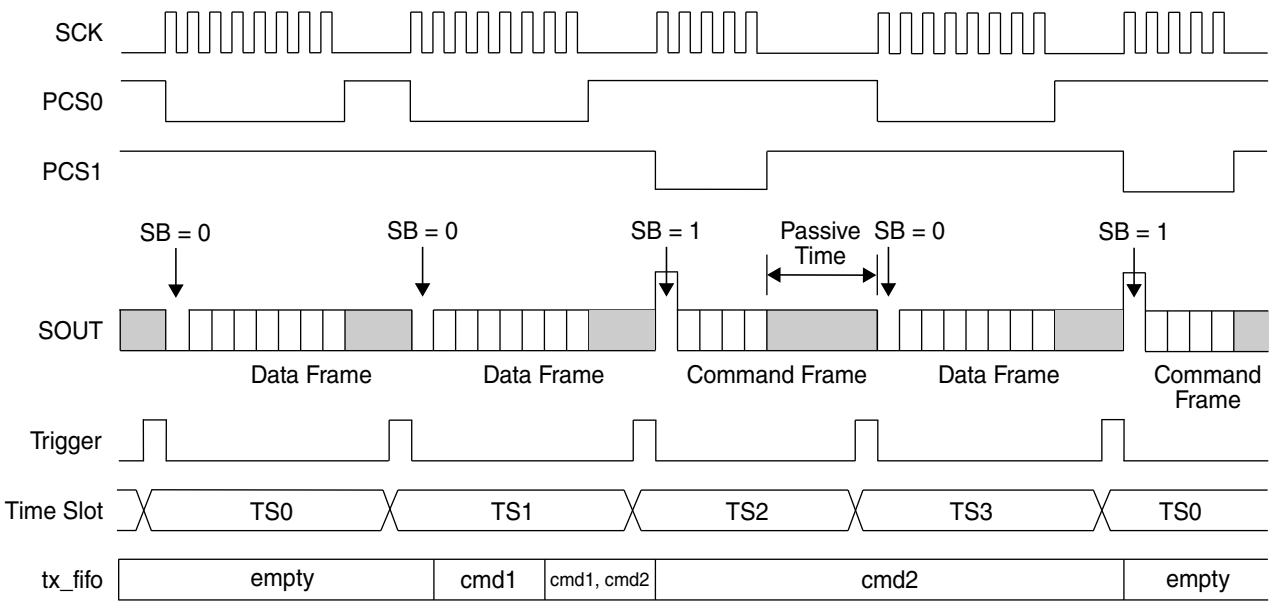


Figure 42-30. Sample MSC Downstream transmission using ITSB mode

Brief description of operation includes:

- At the first trigger, TS0 is initiated. The frame rule for TS0 is configured as CMD_SRDF (Command if valid, else Single Receiver Data Frame). Since there is no data present in the TXFIFO, a valid command frame is not available to be transmitted. Hence a single receiver data frame is transmitted instead.
- The second trigger initiates TS1. Again a single receiver data frame is transmitted since the frame rule for TS1 is programmed to CMD_SRDF and a command frame is not valid as there is no data present in the TXFIFO when TS1 is initiated.
- During TS1, two frames are pushed into the TX_FIFO.
- At third trigger, TS2 is initiated , the frame rule for which is programmed to CMD_PF (Command if valid. else Passive Frame). A Command frame is transmitted as the TX_FIFO is not empty.
    - Since the frame size of the command frame is less than the trigger length for TS2, no transmission happens till next trigger. This is marked as the “Passive Time”. Such passive time exists in all frames after frame transmission completes and until next trigger pulse is detected.

- For TS3, the frame rule programmed is CMD_SRDF. Since the previous transmission was a command frame, a single receiver data frame is transmitted during TS3.
- The time slot TS0 repeats, and this time a command frame is transmitted during TS0 since the Command Frame is eligible to be transmitted.
- At sixth trigger (not shown), TS1 occurs and the process continues.

The figure below shows another sample MSC Downstream transmission for the ITSB mode with the following configuration:

- All TSL[LEN] are programmed to be non-zero. Note that the length between triggers are not drawn to scale in the figure. The lengths of all 4 time slots shown are unequal.
- Data and Command selection bits are enabled, i.e. DSICR1[DSE0] and DSICR1[CSE] are set.
- TS_CONF[TS0] = CMD_DRF1
- TS_CONF[TS1] = DRF2
- TS_CONF[TS2] = CMD_DRF1
- TS_CONF[TS3] = CMD_DRF2
- PCS0 is used for DRF1 while PCS1 is used for DRF2 or Command Frames. Both chip select's are active low.

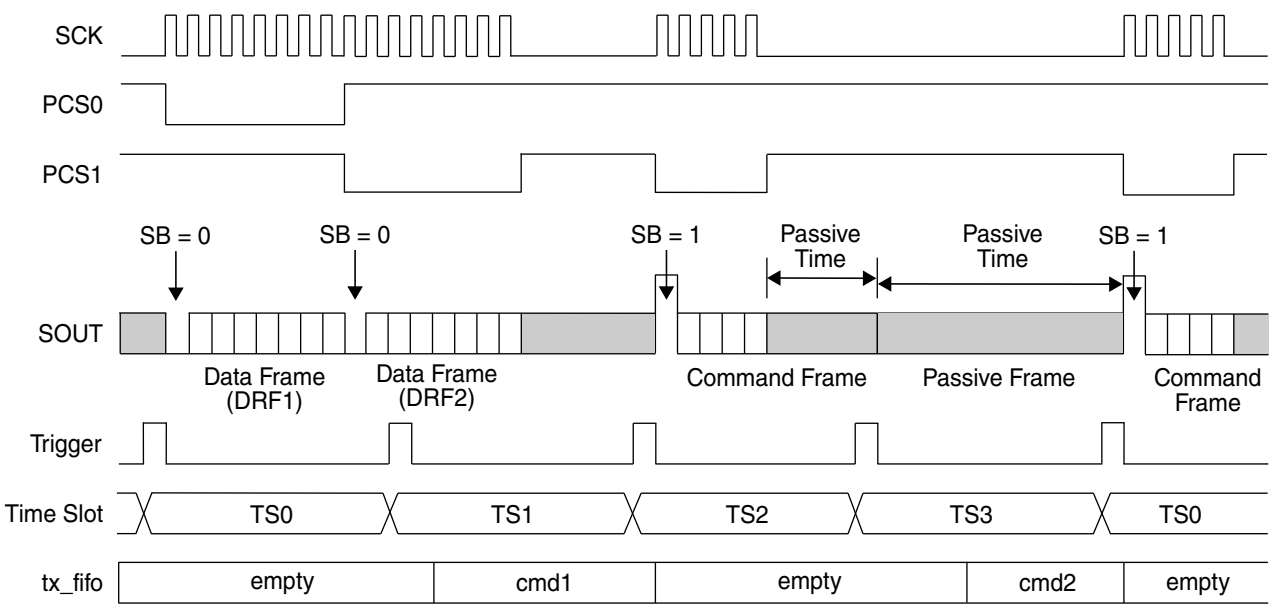


Figure 42-31. Sample MSC Downstream transmission using ITSB mode

Brief description of operation includes:

- At the first trigger, TS0 is initiated. The frame rule for TS0 is configured as CMD_DRF1 (Command if valid, else Dual Receiver Frame 1). Since there is no data present in the TXFIFO, a valid command frame is not available to be transmitted. Hence DRF1 is selected for transmission. Now it is checked if the next trigger is DRF2 which is true and hence DRF1 and DRF2 are transmitted back-to-back seamlessly with the PCS switch-over occurring in between. Note that this happens independent of the next trigger. If the next trigger were configured to CMD_DRF2 instead, then at the current trigger, the validity of CMD is checked to confirm the presence of DRF2 for next trigger. If presence of DRF2 is confirmed for next trigger, it is transmitted along with DRF1 seamlessly as shown here, otherwise only DRF1 would be transmitted in the current trigger and the CMD frame would be transmitted in the next trigger.
- During TS1, one frame is pushed into the TX_FIFO.
- Since DRF2 is transmitted immediately after DRF1, the occurrence second trigger does not cause any operation. Hence it must be ensured by the user that Trigger Length of TS0 must be programmed to minimum possible value - during which DRF1 can be successfully transmitted. This ensures that trigger for TS1 occurs sometime after the switch-over and not after DRF2 has been transmitted.
- The third trigger initiates TS2, the frame rule for which is configured as CMD_DRF1. A Command frame is transmitted as the data required to transmit a command frame is available (TX_FIFO and CMD_FIFO is not empty) and the previous transmitted frame was not a command frame.
- The fourth trigger initiates TS3, the frame rule for which is configured as CMD_DRF2. Since a CMD frame isn't available and DRF2 is not valid (since DRF1 wasn't transmitted in the previous trigger), a passive frame is transmitted instead.
- The time slot TS0 repeats, and this time a command frame is transmitted during TS0 since the Command Frame is eligible to be transmitted.
- At sixth trigger (not shown), TS1 occurs and the process continues.

NOTE

For a command/data frame to be transmitted on a trigger, they should be available some time before the occurrence of the trigger. A DSI frame must be updated/present "X" time units before the occurrence of the trigger for the time slot in which this data is to be transmitted. This is true for all DSI frames and also implies that when DRF1 and DRF2 are positioned to be

transmitted seamlessly, both DRF1 and DRF2 frames must be available 'X' time units before occurrence of trigger pulse for the time slot of DRF1 itself.

Similarly, a command frame must be available "Y" time units before the occurrence of the trigger for the time slot in which this data is to be transmitted.

Here X = 2R + 1P and Y = 1R + 3P; where R = register clock period, P = protocol clock period.

42.5.11 Parity Generation and Check

The module can generate and check parity in the serial frame. The parity bit replaces the last transmitted bit in the frame. The parity is calculated for all transmitted data bits in frame, not including the last data bit that would be transmitted. The parity generation/ control is done on frame basis. The registers field setting frame size defines the total number of bits in the frame, including the parity bit. Thus, to transmit/receive the same number of data bits with parity check, increase the frame size by one versus the same data size frame without the parity check.

Parity can be selected as odd or even. Parity Errors in the received frame set Parity Error flags in the Status register. The Parity Error Interrupt Requests are generated if enabled. The module can be programmed to stop frame transmission in case of a frame reception with parity error.

42.5.11.1 Parity for SPI Frames

When the module is in the master mode the parity generation is controlled by PE and PP bits of the CMD FIFO entries (PUSHR). Setting the PE bit enables parity generation for transmitted SPI frames and parity check for received frames. PP bit defines polarity of the parity bit.

When continuous PCS selection is used to transmit SPI data, two parity generation scenarios are available:

- Generate/check parity for the whole frame
- Generate/check parity for each sub-frame separately.

To generate/check parity for the whole frame set PE bit only in the last command/TX FIFO entry, forming this frame (with the PUSHR register).

To generate/check parity for each sub-frame set PE bit in each command/TX FIFO entry, forming this frame.

If the parity error occurs for received SPI frame, the SR[SPEF] bit is set. If MCR[PES] bit is set, the module stops SPI frames transmission. To resume SPI operation clear the SR[SPEF] or the MCR[PES] bits.

In slave mode the parity is controlled by the PE and PP bits of the CTAR0 register similar to the master mode parity generation without continuous PCS selection.

42.5.11.2 Parity for DSI Frames

When DSPI is in Master Mode, parity generation is controlled by PE and PP bits of the DSPI_DSICR0 register similar to the SPI frames. The parity is calculated and checked for each DSI frame. DSPI_DSICR0[DCONT] bit has no effect on parity generation. In slave mode the parity is controlled by the PE and PP bits of the CTAR1 register.

If the parity error occurs for received DSI frame, the DSPI_SR[DPEF] bit is set. If DSPI_DSICR0[PES] bit is set, the DSPI stops DSI frames transmission. To resume DSI operation clear the DSPI_SR[DPEF] or the DSPI_DSICR0[PES] bits.

42.5.12 Interrupts/DMA requests

The module has several conditions that can generate only interrupt requests and two conditions that can generate interrupt or DMA requests. The following table lists these conditions.

Table 42-16. Interrupt and DMA request conditions

Condition	Flag	Interrupt	DMA
End of Queue (EOQ)	EOQF	Yes	-
TX FIFO Fill	TFFF	Yes	Yes
CMD FIFO Fill	CMDFFF	Yes	Yes
TX FIFO Invalid Write	TFIWF	Yes	-
Transfer Complete	TCF	Yes	-
CMD Transfer Complete	CMDTCF	Yes	-
SPI Transfer Complete	SPITCF	Yes	-
DSI Transfer Complete	DSITCF	Yes	-
TX FIFO Underflow	TFUF	Yes	-
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	-
SPI Parity Error	SPEF	Yes	-

Table continues on the next page...

Table 42-16. Interrupt and DMA request conditions (continued)

Condition	Flag	Interrupt	DMA
DSI Parity Error	DPEF	Yes	-
DSI Deserialized Data Match	DDIF	Yes	Yes

Each condition has a flag bit in the module Status Register (SR) and a Request Enable bit in the DMA/Interrupt Request Select and Enable Register (RSER). Certain flags (as shown in above table) generate interrupt requests or DMA requests depending on configuration of RSER register.

The module also provides a global interrupt request line, which is asserted when any of individual interrupt requests lines is asserted.

42.5.12.1 End Of Queue interrupt request

The End Of Queue (EOQ) interrupt request indicates that the end of a transmit queue is reached. The module generates the interrupt request when EOQ interrupt requests are enabled (RSER[EOQF_RE]) and the EOQ bit in the executing SPI command is 1.

When Extended SPI mode is enabled (MCR[XSPI]) and the EOQ bit in the executing SPI command is 1, the module generates the EOQ interrupt request when the last bit of the last data frame in the command cycle has been transmitted.

When Extended SPI mode is disabled (MCR[XSPI]), the module generates the interrupt request when the last bit of the SPI frame with EOQ bit set is transmitted.

42.5.12.2 Transmit FIFO Fill Interrupt or DMA Request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

NOTE

TFFF flag clears automatically when DMA is used to fill TX FIFO. Configure the DMA to fill only one FIFO location per transfer.

To clear TFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill TX FIFO:

1. Wait until TFFF = 1.
2. Write data to PUSHR using CPU.
3. Clear TFFF by writing a 1 to its location. If TX FIFO is not full, this flag will not clear.

42.5.12.3 Command FIFO Fill Interrupt or DMA Request

The Command FIFO Fill Request indicates that the CMD FIFO is not full. The Command FIFO Fill Request is generated when the number of entries in the CMD FIFO is less than the maximum number of possible entries, and the CMDFFF_RE bit in the RSER is set. The CMDFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

This Request is useful when MCR[XSPI] is enabled, since TX FIFO and CMD FIFO can be filled independantly. If MCR[XSPI] is disabled, then 'TX FIFO Fill Interrupt or DMA Request' will suffice to fill both FIFO's since both FIFO's must be filled simultaneously.

NOTE

CMDFFF flag clears automatically when DMA is used to fill CMD FIFO. Configure the DMA to fill only one FIFO location per transfer.

To clear CMDFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill CMDFIFO:

1. Wait until CMDFFF = 1
2. Write data to PUSHR using CPU.
3. Clear CMDFFF by writing a 1 to its location. If CMD FIFO is not full, this flag will not clear.

42.5.12.4 Transmit FIFO Invalid Write Interrupt Request

The Transmit FIFO Invalid Write Request is valid only when MCR[XSPI] is enabled. This Request indicates that Data exists in the TX FIFO while the CMD FIFO is empty. Since no Command Fields are associated with the Data present in TX FIFO, this data is considered invalid until a Command Entry becomes available. The Transmit FIFO Invalid Write Request is generated for the above condition when TFIWF_RE bit is set in the RSER.

42.5.12.5 Transfer Complete Interrupt Request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the RSER.

42.5.12.6 Command Transfer Complete Interrupt Request

The Command Transfer Complete Request indicates the end of transfer of the last SPI frame in a Command Cycle. The Transfer Complete Request is generated for the above condition when the CMDTCF_RE bit is set in the RSER.

42.5.12.7 SPI Transfer Complete Interrupt Request

The SPI Transfer Complete Request indicates the end of transfer of an SPI frame in any Transmission Mode which uses DSPI in CSI Mode. The SPI Transfer Complete Request is generated when SR[SPITCF] flag asserts and SPITCF_RE bit is set in the RSER.

42.5.12.8 DSI Transfer Complete Interrupt Request

The DSI Transfer Complete Request indicates the end of transfer of a DSI frame in any Transmission Mode which uses DSPI in CSI Mode. The DSI Transfer Complete Request is generated when SR[DSITCF] flag asserts and DSITCF_RE bit is set in the RSER.

42.5.12.9 Transmit FIFO Underflow Interrupt Request

The Transmit FIFO Underflow Request indicates that an underflow condition in the TX FIFO has occurred. The transmit underflow condition is detected only for the module operating in Slave mode and SPI configuration. The TFUF bit is set when the TX FIFO of the module is empty, and a transfer is initiated from an external SPI master. If the TFUF bit is set while the TFUF_RE bit in the RSER is set, an interrupt request is generated.

42.5.12.10 Receive FIFO Drain Interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated. Configure the DMA to drain only one FIFO location per transfer.

42.5.12.11 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

42.5.12.12 SPI Frame Parity Error Interrupt Request

The SPI Frame Parity Error Flag indicates that a SPI frame with parity error had been received. The SPEF_RE bit in the RSER must be set for the interrupt request to be generated.

42.5.12.13 DSI Frame Parity Error Interrupt Request

The DSI Frame Parity Error Flag indicates that a DSI frame with parity error has been received. The DPEF_RE bit in the DSPI_RSER must be set for the interrupt request to be generated.

42.5.12.14 Deserialized Data Match Interrupt or DMA Request

The Deserialized Data Match Flag (DDIF) indicates that a DSI frame matches DSPI_DPIR0–1 data, masked with DSPI_DIMR0–1, had been received. The DDIF_RE bit in the DSPI_RSER must be set for the request to be generated. The DDIF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

42.5.13 Power saving features

The module supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory mapped logic

42.5.13.1 Stop mode (External Stop mode)

This module supports the Stop mode protocol. When a request is made to enter External Stop mode, the module acknowledges the request . If a serial transfer is in progress, then this module waits until it reaches the frame boundary before it is ready to have its clocks shut off . While the clocks are shut off, this module's memory-mapped logic is not accessible. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

42.5.13.2 Module Disable mode

Module Disable mode is a block-specific mode that the module can enter to save power. Host CPU can initiate the Module Disable mode by setting the MDIS bit in the MCR. The Module Disable mode can also be initiated by hardware.

When the MDIS bit is set, the module negates the Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, it is said to have entered Module Disable Mode. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. If implemented, the Clock Enable signal can stop the clock to the non-memory mapped logic. When Clock Enable is negated, the module is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different effect when the module is in the Module Disable mode. Reading the RX FIFO Pop Register does not change the state of the RX FIFO. Similarly, writing to the PUSHR Register does not change the state of the TX FIFO or CMD FIFO. Clearing either of the FIFOs has no effect in the Module Disable mode. Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect in the Module Disable mode. In the Module Disable mode, all status bits and register flags in the module return the correct values when read, but writing to them has no effect. Writing to the TCR during Module Disable mode has no effect. Interrupt and DMA request signals cannot be cleared while in the Module Disable mode.

42.6 Initialization/application information

This section describes how to initialize the module.

42.6.1 How to manage queues

The queues are not part of the module, but it includes features in support of queue management. Queues are primarily supported in SPI configuration.

1. When module executes last command word from a queue, the EOQ bit in the command word is set to indicate it that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the module in the Stopped state. The TXRXS bit is cleared to indicate the Stopped state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO (and CMD FIFO) and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO (and CMD FIFO) by writing a 1 to the CLR_TXF bit in the MCR. Flush RX FIFO by writing a '1' to the CLR_RXF bit in the MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the module TX FIFO, (and CMD FIFO) and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

42.6.2 Switching Master and Slave mode

When changing modes in the module, follow the steps below to guarantee proper operation.

1. Halt it by setting MCR[HALT].
2. Clear the transmit and receive FIFOs by writing a 1 to the CLR_TXF and CLR_RXF bits in MCR.
3. Set the appropriate mode in MCR[MSTR] and enable it by clearing MCR[HALT].

42.6.3 Initializing Module in Master/Slave Modes

Once the appropriate mode in MCR[MSTR] is configured, the module is enabled by clearing MCR[HALT]. It should be ensured that module Slave is enabled before enabling it's Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

42.6.4 Baud rate settings

The following table shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz protocol frequency and the double baud rate DBR bit is cleared.

Table 42-17. Baud rate values (bps)

		Baud rate divider prescaler values			
		2	3	5	7
Baud Rate Scaler Values	2	25.0M	16.7M	10.0M	7.14M
	4	12.5M	8.33M	5.00M	3.57M
	6	8.33M	5.56M	3.33M	2.38M
	8	6.25M	4.17M	2.50M	1.79M
	16	3.12M	2.08M	1.25M	893k
	32	1.56M	1.04M	625k	446k
	64	781k	521k	312k	223k
	128	391k	260k	156k	112k
	256	195k	130k	78.1k	55.8k
	512	97.7k	65.1k	39.1k	27.9k

Table continues on the next page...

Table 42-17. Baud rate values (bps) (continued)

		Baud rate divider prescaler values			
		2	3	5	7
	1024	48.8k	32.6k	19.5k	14.0k
	2048	24.4k	16.3k	9.77k	6.98k
	4096	12.2k	8.14k	4.88k	3.49k
	8192	6.10k	4.07k	2.44k	1.74k
	16384	3.05k	2.04k	1.22k	872
	32768	1.53k	1.02k	610	436

42.6.5 Delay settings

The following table shows the values for the Delay after Transfer (t_{DT}) and CS to SCK Delay (T_{CSC}) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz protocol frequency.

NOTE

The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

This table does not apply for TSB Continuous mode.

Table 42-18. Delay values

		Delay prescaler values			
		1	3	5	7
Delay scaler values	2	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 µs
	32	320.0 ns	960.0 ns	1.6 µs	2.2 µs
	64	640.0 ns	1.9 µs	3.2 µs	4.5 µs
	128	1.3 µs	3.8 µs	6.4 µs	9.0 µs
	256	2.6 µs	7.7 µs	12.8 µs	17.9 µs
	512	5.1 µs	15.4 µs	25.6 µs	35.8 µs
	1024	10.2 µs	30.7 µs	51.2 µs	71.7 µs
	2048	20.5 µs	61.4 µs	102.4 µs	143.4 µs
	4096	41.0 µs	122.9 µs	204.8 µs	286.7 µs
	8192	81.9 µs	245.8 µs	409.6 µs	573.4 µs
	16384	163.8 µs	491.5 µs	819.2 µs	1.1 ms

Table continues on the next page...

Table 42-18. Delay values (continued)

		Delay prescaler values			
		1	3	5	7
	32768	327.7 µs	983.0 µs	1.6 ms	2.3 ms
	65536	655.4 µs	2.0 ms	3.3 ms	4.6 ms

42.6.6 Calculation of FIFO pointer addresses

Complete visibility of the FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and counter for each FIFO. The pointer to the first-in entry in each FIFO is memory mapped. For the TX FIFO the first-in pointer is the Transmit Next Pointer (TXNXTPTR). For the CMD FIFO the first-in pointer is the Command Next Pointer (CMDNXTPTR). For the RX FIFO the first-in pointer is the Pop Next Pointer (POPNXTPTR). The following figure illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts carry over. See Transmit First In First Out (TX FIFO) buffering mechanism, Command First In First Out (CMD FIFO) Buffering Mechanism and Receive First In First Out (RX FIFO) buffering mechanism for details on the FIFO operation.

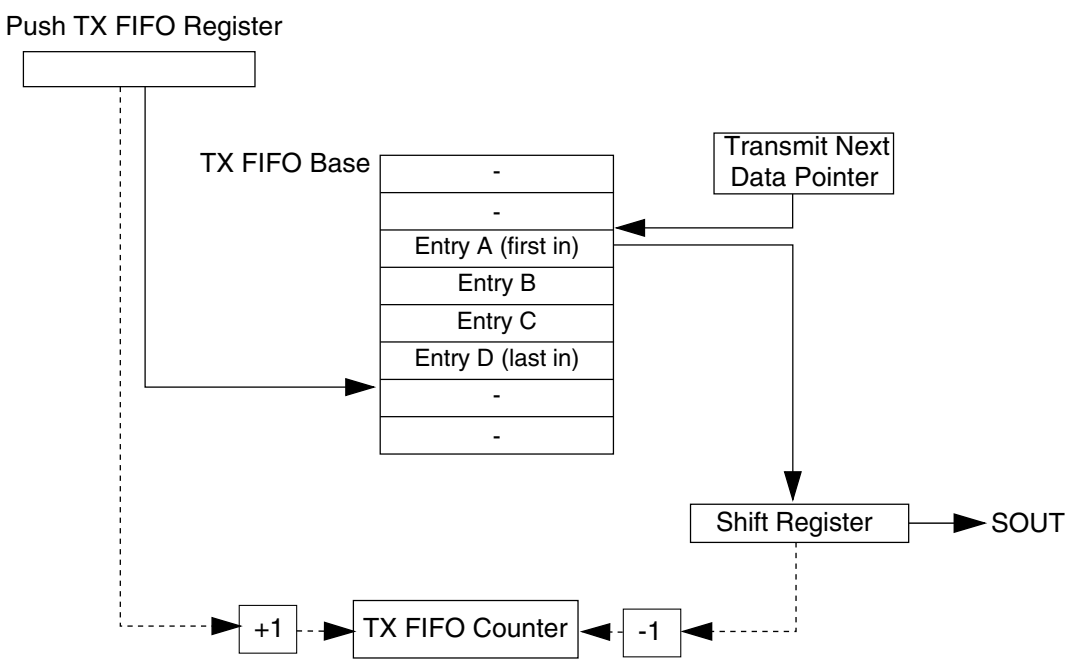


Figure 42-32. TX FIFO pointers and counter

42.6.6.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + (4 \times \text{TXNXTPTR})$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times (\text{TXCTR} + \text{TXNXTPTR} - 1) \bmod (\text{TXFIFOdepth})$$

TX FIFO Base - Base address of TX FIFO
TXCTR - TX FIFO Counter
TXNXTPTR - Transmit Next Pointer
TX FIFO Depth - Transmit FIFO depth, implementation specific

42.6.6.2 Address Calculation for the First-in Entry and Last-in Entry in the CMD FIFO

The memory address of the first-in entry in the CMD FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + (4 \times \text{TXNXTPTR})$$

The memory address of the last-in entry in the CMD FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times (\text{TXCTR} + \text{TXNXTPTR} - 1) \bmod (\text{TXFIFOdepth})$$

CMD FIFO Base - Base address of CMD FIFO
CMDCTR - CMD FIFO Counter
CMDNXTPTR - Command Next Pointer
CMD FIFO Depth - Command FIFO depth, implementation specific

42.6.6.3 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{RX FIFOBase} + \left(4 \times \text{POPNXTPTR}\right)$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{RX FIFO Base} + 4 \times \left(\text{RXCTR} + \text{POPNXTPTR} - 1\right) \text{mod} \left(\text{RXFIFOdepth}\right)$$

RX FIFO Base - Base address of RX FIFO
RXCTR - RX FIFO counter
POPNXTPTR - Pop Next Pointer
RX FIFO Depth - Receive FIFO depth, implementation specific

Chapter 43
Ethernet MAC (ENET)

43.1 Chip-specific ENET information

43.1.1 ENET Configuration

NOTE

It is only possible to use ENET0 and ENET1 simultaneously when both are configured for MII_Lite

ENET byte swapping control

The endianess of the ENET AHB interface is controlled via the GPR register bits in the device.

Refer -GPR Control Register (GPR_CTL) and -GPR Status Register (GPR_STAT) for details.

43.1.2 ENET Interrupt mapping

The following tables lists the ENET interrupt mapping implemented in this device.

Table 43-1. ENET0 Interrupt mapping

ENET0 Interrupt Group0	MAC 0 Periodic Timer Overflow
	MAC 0 Time Stamp Available
	MAC 0 Wakeup Request (sync)
	MAC 0 Payload Receive Error
	MAC 0 Transmit FIFO Underrun
	MAC 0 Collision Retry Limit
	MAC 0 Late Collision
	MAC 0 Ethernet Bus Error

Table continues on the next page...

Table 43-1. ENET0 Interrupt mapping (continued)

	MAC 0 MII Data Transfer Done
	MAC 0 Graceful Stop
	MAC 0 Babbling Transmit Error
	MAC 0 Babbling Receive Error
ENET0 Interrupt Group1	RX_FLUSH0
	MAC 0 Receive Buffer Done
	MAC 0 Receive Frame Done
ENET0 Interrupt Group2	MAC 0 Transmit Buffer Done
	MAC 0 Transmit Frame Done
ENET0 Interrupt Group3	MAC 0 Frame rejected due to Match/Inverse Match
	MAC 0 Frame rejected due to Parser Error
ENET0 Interrupt Group4	MAC 0 1588 Timer Interrupt - synchronous
ENET0 Interrupt Group5	RX_FLUSH1
	Receive Buffer_1 Done
	Receive Frame_1 Done
ENET0 Interrupt Group6	Transmit Buffer_1 Done
	Transmit Frame_1 Done
ENET0 Interrupt Group7	RX_FLUSH2
	Receive Buffer_2 Done
	Receive Frame_2 Done
ENET0 Interrupt Group8	Transmit Buffer_2 Done
	Transmit Frame_2 Done

43.1.3 ENET's stop-stop ack protocol

1. An acknowledgment to stop request is sent only when the MAC is disabled or in the idle state.
2. If the stop request is asserted to the ENET, only magic packet detection can wakeup the module.

43.1.4 Recieve Parser Match Array Table Depth

The depth of Physical table for Match Array table, which is the maximum value that can be programmed in ENET_RXPCTL[MAXINDEX] register field is 64.

43.2 Introduction

The MAC-NET core, in conjunction with a 10/100-Mbit/s MAC, implements layer 3 network acceleration functions. These functions are designed to accelerate the processing of various common networking protocols, such as IP, TCP, UDP, and ICMP, providing wire speed services to client applications.

43.3 Overview

The core implements a dual-speed 10/100-Mbit/s Ethernet MAC compliant with the IEEE802.3-2002 standard. The MAC layer provides compatibility with half- or full-duplex 10/100-Mbit/s Ethernet LANs.

The MAC operation is fully programmable and can be used in Network Interface Card (NIC), bridging, or switching applications. The core implements the remote network monitoring (RMON) counters according to IETF RFC 2819.

The core also implements a hardware acceleration block to optimize the performance of network controllers providing TCP/IP, UDP, and ICMP protocol services. The acceleration block performs critical functions in hardware, which are typically implemented with large software overhead.

The core implements programmable embedded FIFOs that can provide buffering on the receive path for lossless flow control.

Advanced power management features are available with magic packet detection and programmable power-down modes.

A unified DMA (uDMA), internal to the ENET module, optimizes data transfer between the ENET core and the SoC, and supports an enhanced buffer descriptor programming model to support IEEE 1588 functionality.

The programmable Ethernet MAC with IEEE 1588 integrates a standard IEEE 802.3 Ethernet MAC with a time-stamping module. The IEEE 1588 standard provides accurate clock synchronization for distributed control nodes for industrial automation applications.

43.3.1 Features

The MAC-NET core includes the following features.

43.3.1.1 Ethernet MAC features

- Implements the full 802.3 specification with preamble/SFD generation, frame padding generation, CRC generation and checking
- Supports zero-length preamble
- Dynamically configurable to support 10/100-Mbit/s operation
- Supports 10/100 Mbit/s full-duplex and configurable half-duplex operation
- Compliant with the AMD magic packet detection with interrupt for node remote power management
- Seamless interface to commercial ethernet PHY devices via one of the following:
  - a 4-bit Media Independent Interface (MII) operating at 2.5/25 MHz.
  - a 4-bit non-standard MII-Lite (MII without the CRS and COL signals) operating at 2.5/25 MHz.
  - a 2-bit Reduced MII (RMII) operating at 50 MHz.
- Simple 64-Bit FIFO user-application interface
- CRC-32 checking at full speed with optional forwarding of the frame check sequence (FCS) field to the client
- CRC-32 generation and append on transmit or forwarding of user application provided FCS selectable on a per-frame basis
- In full-duplex mode:
  - Implements automated pause frame (802.3 x31A) generation and termination, providing flow control without user application intervention
  - Pause quanta used to form pause frames — dynamically programmable
  - Pause frame generation additionally controllable by user application offering flexible traffic flow control
  - Optional forwarding of received pause frames to the user application
  - Implements standard flow-control mechanism
- In half-duplex mode: provides full collision support, including jamming, backoff, and automatic retransmission
- Supports VLAN-tagged frames according to IEEE 802.1Q
- Programmable MAC address: Insertion on transmit; discards frames with mismatching destination address on receive (except broadcast and pause frames)
- Programmable promiscuous mode support to omit MAC destination address checking on receive
- Multicast and unicast address filtering on receive based on 64-entry hash table, reducing higher layer processing load
- Programmable frame maximum length providing support for any standard or proprietary frame length

- Statistics indicators for frame traffic and errors (alignment, CRC, length) and pause frames providing for IEEE 802.3 basic and mandatory management information database (MIB) package and remote network monitoring (RFC 2819)
- Simple handshake user application FIFO interface with fully programmable depth and threshold levels
- Provides separate status word for each received frame on the user interface providing information such as frame length, frame type, VLAN tag, and error information
- Multiple internal loopback options
- MDIO master interface for PHY device configuration and management supports two programmable MDIO base addresses, and standard (IEEE 802.3 Clause 22) and extended (Clause 45) MDIO frame formats
- Supports legacy FEC buffer descriptors
- Interrupt coalescing reduces the number of interrupts generated by the MAC, reducing CPU loading
- Traffic-shaping bandwidth distribution supports credit-based and round-robin-based policies. Either policy can be combined with time-based shaping.
- AVB (Audio Video Bridging, IEEE 802.1Qav) features:
  - Credit-based bandwidth distribution policy can be combined with time-based shaping
  - AVB endpoint talker and listener support
  - Support for arbitration between different priority traffic (for example, AVB class A, AVB class B, and non-AVB)
- Receive frame parser enables flexible Ethernet frame pattern matching in order to finally accept or reject a frame.

43.3.1.2 IP protocol performance optimization features

- Operates on TCP/IP and UDP/IP and ICMP/IP protocol data or IP header only
- Enables wire-speed processing
- Supports IPv4 and IPv6
- Transparent passing of frames of other types and protocols
- Supports VLAN tagged frames according to IEEE 802.1q with transparent forwarding of VLAN tag and control field
- Automatic IP-header and payload (protocol specific) checksum calculation and verification on receive
- Automatic IP-header and payload (protocol specific) checksum generation and automatic insertion on transmit configurable on a per-frame basis

- Supports IP and TCP, UDP, ICMP data for checksum generation and checking
- Supports full header options for IPv4 and TCP protocol headers
- Provides IPv6 support to datagrams with base header only — datagrams with extension headers are passed transparently unmodifed/unchecked
- Provides statistics information for received IP and protocol errors
- Configurable automatic discard of erroneous frames
- Configurable automatic host-to-network (RX) and network-to-host (TX) byte order conversion for IP and TCP/UDP/ICMP headers within the frame
- Configurable padding remove for short IP datagrams on receive
- Configurable Ethernet payload alignment to allow for 32-bit word-aligned header and payload processing
- Programmable store-and-forward operation with clock and rate decoupling FIFOs

43.3.1.3 IEEE 1588 features

- Supports all IEEE 1588 frames.
- Allows reference clock to be chosen independently of network speed.
- Software-programmable precise time-stamping of ingress and egress frames
- Timer monitoring capabilities for system calibration and timing accuracy management
- Precise time-stamping of external events with programmable interrupt generation
- Programmable event and interrupt generation for external system control
- Supports hardware- and software-controllable timer synchronization.
- Provides a 4-channel IEEE 1588 timer. Each channel supports input capture and output compare using the 1588 counter.

43.3.2 Block diagram

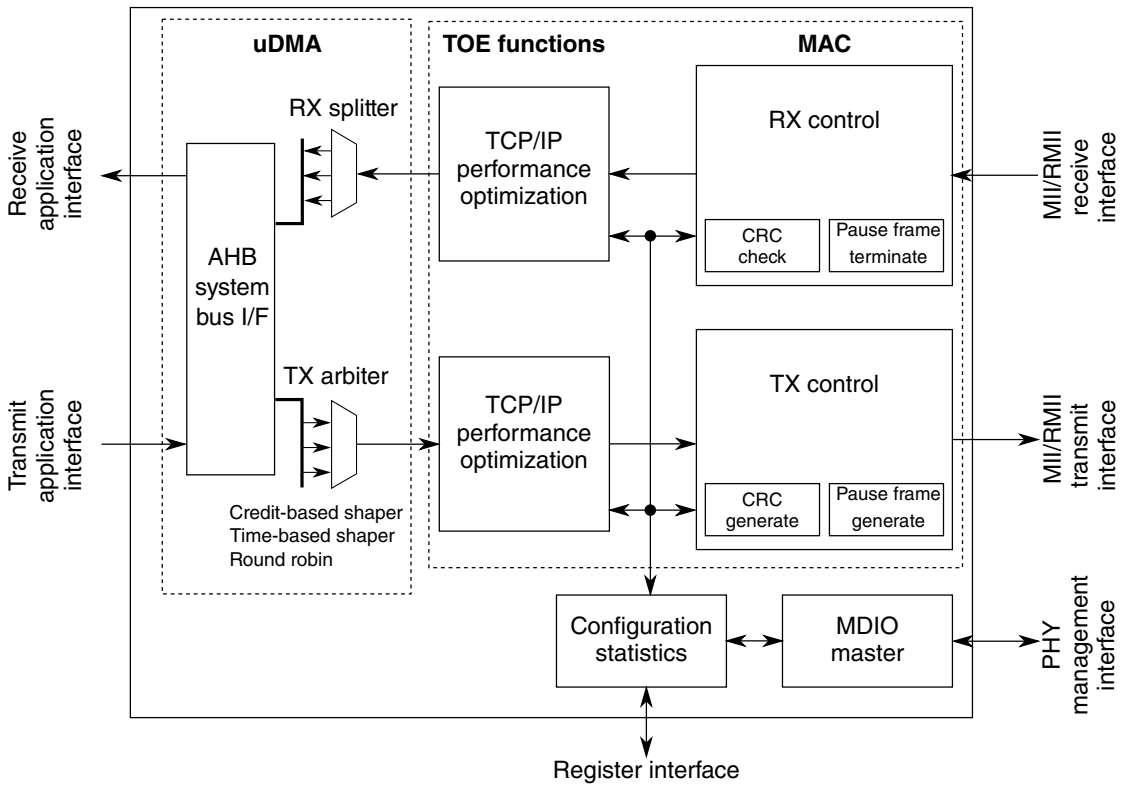


Figure 43-1. Ethernet MAC-NET core block diagram

43.4 External signal description

NOTE

The MII column pertains only to devices that support MII.

MII	RMII	Description	I/O
MII_COL	—	Asserted upon detection of a collision and remains asserted while the collision persists. This signal is not defined for full-duplex mode.	I
MII_CRS	—	Carrier sense. When asserted, indicates transmit or receive medium is not idle. In RMII mode, this signal is present on the RMII_CRS_DV pin.	I

Table continues on the next page...

MII	RMII	Description	I/O
MII_MDC	RMII_MDC	Output clock provides a timing reference to the PHY for data transfers on the MDIO signal.	O
MII_MDIO	RMII_MDIO	Transfers control information between the external PHY and the media-access controller. Data is synchronous to MDC. This signal is an input after reset.	I/O
MII_RXCLK	—	In MII mode, provides a timing reference for RXDV, RXD[3:0], and RXER.	I
MII_RXDV	RMII_CRS_DV	Asserting this input indicates the PHY has valid nibbles present on the MII. RXDV must remain asserted from the first recovered nibble of the frame through to the last nibble. Asserting RXDV must start no later than the SFD and exclude any EOF. In RMII mode, this pin also generates the CRS signal.	I
MII_RXD[3:0]	RMII_RXD[1:0]	Contains the Ethernet input data transferred from the PHY to the media-access controller when RXDV is asserted.	I
MII_RXER	RMII_RXER	When asserted with RXDV, indicates the PHY detects an error in the current frame.	I
MII_TXCLK	—	Input clock, which provides a timing reference for TXEN, TXD[3:0], and TXER.	I
MII_TXD[3:0]	RMII_TXD[1:0]	Serial output Ethernet data. Only valid during TXEN assertion.	O
MII_TXEN	RMII_TXEN	Indicates when valid nibbles are present on the MII. This signal is asserted with the first nibble of a preamble and is deasserted before the first TXCLK following the final nibble of the frame.	O
MII_TXER	—	When asserted for one or more clock cycles while TXEN is also asserted, PHY sends one or more illegal symbols.	O

Table continues on the next page...

MII	RMII	Description	I/O
—	RMII_REF_CLK	In RMII mode, this signal is the reference clock for receive, transmit, and the control interface.	I
1588_TMR*n*	1588_TMR*n*	Capture/Compare block input/output event bus. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCR*n* register for inspection by software. When configured for compare, the corresponding signal 1588_TMR*n* is asserted for 1 to 32 cycles when the timer reaches the compare value programmed in ENET_TCCR*n*. An interrupt can be triggered if ENET_TCSR*n*[TIE] is set. A DMA request can be triggered if ENET_TCSR*n*[TDRE] is set.	I/O
ENET_1588_CLKIN	ENET_1588_CLKIN	Alternate IEEE 1588 Ethernet clock input; Clock period should be an integer number of nanoseconds	I

43.5 Memory map/register definition

ENET registers must be read or written with 32-bit accesses. Non-32 bit accesses will terminate with an error.

Reserved bits should be written with 0 and ignored on read. Unused registers read zero and a write has no effect.

This table shows Ethernet registers organization.

Table 43-2. Register map summary

Offset Address	Section	Description
0x0000 – 0x01FF	Configuration	Core control and status registers
0x0200 – 0x03FF	Statistics counters	MIB and Remote Network Monitoring (RFC 2819) registers
0x0400 – 0x0430	1588 control	1588 adjustable timer (TSM) and 1588 frame control
0x0580 – 0x05FF	Receive Parser	Receive parser configuration

Table continues on the next page...

Table 43-2. Register map summary (continued)

Offset Address	Section	Description
0x0600 – 0x07FC	Capture/Compare block	Registers for the Capture/Compare block

ENET memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4	Interrupt Event Register (ENET_EIR)	32	w1c	0000_0000h	43.5.1/1362
8	Interrupt Mask Register (ENET_EIMR)	32	R/W	0000_0000h	43.5.2/1365
10	Receive Descriptor Active Register - Ring 0 (ENET_RDAR)	32	R/W	0000_0000h	43.5.3/1369
14	Transmit Descriptor Active Register - Ring 0 (ENET_TDAR)	32	R/W	0000_0000h	43.5.4/1370
24	Ethernet Control Register (ENET_ECR)	32	R/W	See section	43.5.5/1371
40	MII Management Frame Register (ENET_MMFR)	32	R/W	0000_0000h	43.5.6/1373
44	MII Speed Control Register (ENET_MSCR)	32	R/W	0000_0000h	43.5.7/1374
64	MIB Control Register (ENET_MIBC)	32	R/W	C000_0000h	43.5.8/1376
84	Receive Control Register (ENET_RCR)	32	R/W	05EE_0001h	43.5.9/1377
C4	Transmit Control Register (ENET_TCR)	32	R/W	0000_0000h	43.5.10/ 1380
E4	Physical Address Lower Register (ENET_PALR)	32	R/W	0000_0000h	43.5.11/ 1382
E8	Physical Address Upper Register (ENET_PAUR)	32	R/W	0000_8808h	43.5.12/ 1382
EC	Opcode/Pause Duration Register (ENET_OPD)	32	R/W	0001_0000h	43.5.13/ 1383
F0	Transmit Interrupt Coalescing Register (ENET_TXIC0)	32	R/W	0000_0000h	43.5.14/ 1383
F4	Transmit Interrupt Coalescing Register (ENET_TXIC1)	32	R/W	0000_0000h	43.5.14/ 1383
F8	Transmit Interrupt Coalescing Register (ENET_TXIC2)	32	R/W	0000_0000h	43.5.14/ 1383
100	Receive Interrupt Coalescing Register (ENET_RXIC0)	32	R/W	0000_0000h	43.5.15/ 1384
104	Receive Interrupt Coalescing Register (ENET_RXIC1)	32	R/W	0000_0000h	43.5.15/ 1384
108	Receive Interrupt Coalescing Register (ENET_RXIC2)	32	R/W	0000_0000h	43.5.15/ 1384
118	Descriptor Individual Upper Address Register (ENET_IAUR)	32	R/W	0000_0000h	43.5.16/ 1385
11C	Descriptor Individual Lower Address Register (ENET_IALR)	32	R/W	0000_0000h	43.5.17/ 1386
120	Descriptor Group Upper Address Register (ENET_GAUR)	32	R/W	0000_0000h	43.5.18/ 1386
124	Descriptor Group Lower Address Register (ENET_GALR)	32	R/W	0000_0000h	43.5.19/ 1387

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
144	Transmit FIFO Watermark Register (ENET_TFWR)	32	R/W	0000_0000h	43.5.20/ 1387
160	Receive Descriptor Ring 1 Start Register (ENET_RDSR1)	32	R/W	0000_0000h	43.5.21/ 1388
164	Transmit Buffer Descriptor Ring 1 Start Register (ENET_TDSR1)	32	R/W	0000_0000h	43.5.22/ 1389
168	Maximum Receive Buffer Size Register - Ring 1 (ENET_MRBR1)	32	R/W	0000_0000h	43.5.23/ 1390
16C	Receive Descriptor Ring 2 Start Register (ENET_RDSR2)	32	R/W	0000_0000h	43.5.24/ 1391
170	Transmit Buffer Descriptor Ring 2 Start Register (ENET_TDSR2)	32	R/W	0000_0000h	43.5.25/ 1391
174	Maximum Receive Buffer Size Register - Ring 2 (ENET_MRBR2)	32	R/W	0000_0000h	43.5.26/ 1392
180	Receive Descriptor Ring 0 Start Register (ENET_RDSR)	32	R/W	0000_0000h	43.5.27/ 1393
184	Transmit Buffer Descriptor Ring 0 Start Register (ENET_TDSR)	32	R/W	0000_0000h	43.5.28/ 1394
188	Maximum Receive Buffer Size Register - Ring 0 (ENET_MRBR)	32	R/W	0000_0000h	43.5.29/ 1394
190	Receive FIFO Section Full Threshold (ENET_RSFL)	32	R/W	0000_0000h	43.5.30/ 1395
194	Receive FIFO Section Empty Threshold (ENET_RSEM)	32	R/W	0000_0000h	43.5.31/ 1396
198	Receive FIFO Almost Empty Threshold (ENET_RAEM)	32	R/W	0000_0004h	43.5.32/ 1396
19C	Receive FIFO Almost Full Threshold (ENET_RAFL)	32	R/W	0000_0004h	43.5.33/ 1397
1A0	Transmit FIFO Section Empty Threshold (ENET_TSEM)	32	R/W	0000_0000h	43.5.34/ 1397
1A4	Transmit FIFO Almost Empty Threshold (ENET_TAEM)	32	R/W	0000_0004h	43.5.35/ 1398
1A8	Transmit FIFO Almost Full Threshold (ENET_TAFL)	32	R/W	0000_0008h	43.5.36/ 1398
1AC	Transmit Inter-Packet Gap (ENET_TIPG)	32	R/W	0000_000Ch	43.5.37/ 1399
1B0	Frame Truncation Length (ENET_FTRL)	32	R/W	0000_07FFh	43.5.38/ 1399
1C0	Transmit Accelerator Function Configuration (ENET_TACC)	32	R/W	0000_0000h	43.5.39/ 1400
1C4	Receive Accelerator Function Configuration (ENET_RACC)	32	R/W	0000_0000h	43.5.40/ 1401
1C8	Receive Classification Match Register for Class n (ENET_RCMR1)	32	R/W	0000_0000h	43.5.41/ 1403

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1CC	Receive Classification Match Register for Class n (ENET_RCMR2)	32	R/W	0000_0000h	43.5.41/ 1403
1D8	DMA Class Based Configuration (ENET_DMA1CFG)	32	R/W	0000_0000h	43.5.42/ 1404
1DC	DMA Class Based Configuration (ENET_DMA2CFG)	32	R/W	0000_0000h	43.5.42/ 1404
1E0	Receive Descriptor Active Register - Ring 1 (ENET_RDAR1)	32	R/W	0000_0000h	43.5.43/ 1406
1E4	Transmit Descriptor Active Register - Ring 1 (ENET_TDAR1)	32	R/W	0000_0000h	43.5.44/ 1406
1E8	Receive Descriptor Active Register - Ring 2 (ENET_RDAR2)	32	R/W	0000_0000h	43.5.45/ 1408
1EC	Transmit Descriptor Active Register - Ring 2 (ENET_TDAR2)	32	R/W	0000_0000h	43.5.46/ 1409
1F0	QOS Scheme (ENET_QOS)	32	R/W	0000_0000h	43.5.47/ 1409
200	Reserved Statistic Register (ENET_RMON_T_DROP)	32	R	0000_0000h	43.5.48/ 1411
204	Tx Packet Count Statistic Register (ENET_RMON_T_PACKETS)	32	R	0000_0000h	43.5.49/ 1411
208	Tx Broadcast Packets Statistic Register (ENET_RMON_T_BC_PKT)	32	R	0000_0000h	43.5.50/ 1412
20C	Tx Multicast Packets Statistic Register (ENET_RMON_T_MC_PKT)	32	R	0000_0000h	43.5.51/ 1412
210	Tx Packets with CRC/Align Error Statistic Register (ENET_RMON_T_CRC_ALIGN)	32	R	0000_0000h	43.5.52/ 1413
214	Tx Packets Less Than Bytes and Good CRC Statistic Register (ENET_RMON_T_UNDERSIZE)	32	R	0000_0000h	43.5.53/ 1413
218	Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (ENET_RMON_T_OVERSIZE)	32	R	0000_0000h	43.5.54/ 1414
21C	Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_T_FRAG)	32	R	0000_0000h	43.5.55/ 1414
220	Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (ENET_RMON_T_JAB)	32	R	0000_0000h	43.5.56/ 1415
224	Tx Collision Count Statistic Register (ENET_RMON_T_COL)	32	R	0000_0000h	43.5.57/ 1415
228	Tx 64-Byte Packets Statistic Register (ENET_RMON_T_P64)	32	R	0000_0000h	43.5.58/ 1416
22C	Tx 65- to 127-byte Packets Statistic Register (ENET_RMON_T_P65TO127)	32	R	0000_0000h	43.5.59/ 1416
230	Tx 128- to 255-byte Packets Statistic Register (ENET_RMON_T_P128TO255)	32	R	0000_0000h	43.5.60/ 1417
234	Tx 256- to 511-byte Packets Statistic Register (ENET_RMON_T_P256TO511)	32	R	0000_0000h	43.5.61/ 1417

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
238	Tx 512- to 1023-byte Packets Statistic Register (ENET_RMON_T_P512TO1023)	32	R	0000_0000h	43.5.62/ 1418
23C	Tx 1024- to 2047-byte Packets Statistic Register (ENET_RMON_T_P1024TO2047)	32	R	0000_0000h	43.5.63/ 1418
240	Tx Packets Greater Than 2048 Bytes Statistic Register (ENET_RMON_T_P_GTE2048)	32	R	0000_0000h	43.5.64/ 1419
244	Tx Octets Statistic Register (ENET_RMON_T_OCTETS)	32	R	0000_0000h	43.5.65/ 1419
248	Reserved Statistic Register (ENET_IEEE_T_DROP)	32	R	0000_0000h	43.5.66/ 1419
24C	Frames Transmitted OK Statistic Register (ENET_IEEE_T_FRAME_OK)	32	R	0000_0000h	43.5.67/ 1420
250	Frames Transmitted with Single Collision Statistic Register (ENET_IEEE_T_1COL)	32	R	0000_0000h	43.5.68/ 1420
254	Frames Transmitted with Multiple Collisions Statistic Register (ENET_IEEE_T_MCOL)	32	R	0000_0000h	43.5.69/ 1421
258	Frames Transmitted after Deferral Delay Statistic Register (ENET_IEEE_T_DEF)	32	R	0000_0000h	43.5.70/ 1421
25C	Frames Transmitted with Late Collision Statistic Register (ENET_IEEE_T_LCOL)	32	R	0000_0000h	43.5.71/ 1422
260	Frames Transmitted with Excessive Collisions Statistic Register (ENET_IEEE_T_EXCOL)	32	R	0000_0000h	43.5.72/ 1422
264	Frames Transmitted with Tx FIFO Underrun Statistic Register (ENET_IEEE_T_MACERR)	32	R	0000_0000h	43.5.73/ 1423
268	Frames Transmitted with Carrier Sense Error Statistic Register (ENET_IEEE_T_CSERR)	32	R	0000_0000h	43.5.74/ 1423
26C	Reserved Statistic Register (ENET_IEEE_T_SQE)	32	R (reads 0)	0000_0000h	43.5.75/ 1424
270	Flow Control Pause Frames Transmitted Statistic Register (ENET_IEEE_T_FDXFC)	32	R	0000_0000h	43.5.76/ 1424
274	Octet Count for Frames Transmitted w/o Error Statistic Register (ENET_IEEE_T_OCTETS_OK)	32	R	0000_0000h	43.5.77/ 1425
284	Rx Packet Count Statistic Register (ENET_RMON_R_PACKETS)	32	R	0000_0000h	43.5.78/ 1425
288	Rx Broadcast Packets Statistic Register (ENET_RMON_R_BC_PKT)	32	R	0000_0000h	43.5.79/ 1426
28C	Rx Multicast Packets Statistic Register (ENET_RMON_R_MC_PKT)	32	R	0000_0000h	43.5.80/ 1426
290	Rx Packets with CRC/Align Error Statistic Register (ENET_RMON_R_CRC_ALIGN)	32	R	0000_0000h	43.5.81/ 1427
294	Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (ENET_RMON_R_UNDERSIZE)	32	R	0000_0000h	43.5.82/ 1427
298	Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (ENET_RMON_R_OVERSIZE)	32	R	0000_0000h	43.5.83/ 1428

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
29C	Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_R_FRAG)	32	R	0000_0000h	43.5.84/ 1428
2A0	Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (ENET_RMON_R_JAB)	32	R	0000_0000h	43.5.85/ 1429
2A4	Reserved Statistic Register (ENET_RMON_R_RESVD_0)	32	R (reads 0)	0000_0000h	43.5.86/ 1429
2A8	Rx 64-Byte Packets Statistic Register (ENET_RMON_R_P64)	32	R	0000_0000h	43.5.87/ 1429
2AC	Rx 65- to 127-Byte Packets Statistic Register (ENET_RMON_R_P65TO127)	32	R	0000_0000h	43.5.88/ 1430
2B0	Rx 128- to 255-Byte Packets Statistic Register (ENET_RMON_R_P128TO255)	32	R	0000_0000h	43.5.89/ 1430
2B4	Rx 256- to 511-Byte Packets Statistic Register (ENET_RMON_R_P256TO511)	32	R	0000_0000h	43.5.90/ 1431
2B8	Rx 512- to 1023-Byte Packets Statistic Register (ENET_RMON_R_P512TO1023)	32	R	0000_0000h	43.5.91/ 1431
2BC	Rx 1024- to 2047-Byte Packets Statistic Register (ENET_RMON_R_P1024TO2047)	32	R	0000_0000h	43.5.92/ 1432
2C0	Rx Packets Greater than 2048 Bytes Statistic Register (ENET_RMON_R_P_GTE2048)	32	R	0000_0000h	43.5.93/ 1432
2C4	Rx Octets Statistic Register (ENET_RMON_R_OCTETS)	32	R	0000_0000h	43.5.94/ 1433
2C8	Frames not Counted Correctly Statistic Register (ENET_IEEE_R_DROP)	32	R	0000_0000h	43.5.95/ 1433
2CC	Frames Received OK Statistic Register (ENET_IEEE_R_FRAME_OK)	32	R	0000_0000h	43.5.96/ 1434
2D0	Frames Received with CRC Error Statistic Register (ENET_IEEE_R_CRC)	32	R	0000_0000h	43.5.97/ 1434
2D4	Frames Received with Alignment Error Statistic Register (ENET_IEEE_R_ALIGN)	32	R	0000_0000h	43.5.98/ 1435
2D8	Receive FIFO Overflow Count Statistic Register (ENET_IEEE_R_MACERR)	32	R	0000_0000h	43.5.99/ 1435
2DC	Flow Control Pause Frames Received Statistic Register (ENET_IEEE_R_FDXFC)	32	R	0000_0000h	43.5.100/ 1436
2E0	Octet Count for Frames Received without Error Statistic Register (ENET_IEEE_R_OCTETS_OK)	32	R	0000_0000h	43.5.101/ 1436
400	Adjustable Timer Control Register (ENET_ATCR)	32	R/W	0000_0000h	43.5.102/ 1436
404	Timer Value Register (ENET_ATVR)	32	R/W	0000_0000h	43.5.103/ 1438
408	Timer Offset Register (ENET_ATOFF)	32	R/W	0000_0000h	43.5.104/ 1439
40C	Timer Period Register (ENET_ATPER)	32	R/W	3B9A_CA00h	43.5.105/ 1439

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
410	Timer Correction Register (ENET_ATCOR)	32	R/W	0000_0000h	43.5.106/ 1440
414	Time-Stamping Clock Period Register (ENET_ATINC)	32	R/W	0000_0000h	43.5.107/ 1440
418	Timestamp of Last Transmitted Frame (ENET_ATSTMP)	32	R	0000_0000h	43.5.108/ 1441
580	Pattern Match Data Register (ENET_MDATA)	32	R/W	0000_0000h	43.5.109/ 1441
584	Match Entry Mask Register (ENET_MMASK)	32	R/W	0000_0000h	43.5.110/ 1442
588	Match Entry Rules Configuration Register (ENET_MCONFIG)	32	R/W	0000_0000h	43.5.111/ 1442
58C	Match Entry Read/Write Command Register (ENET_MENTRYRW)	32	R/W	0000_0000h	43.5.112/ 1443
590	Receive Parser Control Register (ENET_RXPCTL)	32	R/W	0000_0000h	43.5.113/ 1444
594	Maximum Frame Offset (ENET_MAXFRMOFF)	32	R/W	0000_001Fh	43.5.114/ 1446
598	Receive Parser Status (ENET_RXPARST)	32	R	See section	43.5.115/ 1447
5A0	Parser Discard Count (ENET_PARSDSCD)	32	R	0000_0000h	43.5.116/ 1448
5A4	Parser Accept Count 0 (ENET_PRSACPT0)	32	R	0000_0000h	43.5.117/ 1449
5A8	Parser Reject Count 0 (ENET_PRSRJCT0)	32	R	0000_0000h	43.5.118/ 1449
5AC	Parser Accept Count 1 (ENET_PRSACPT1)	32	R	0000_0000h	43.5.119/ 1450
5B0	Parser Reject Count 1 (ENET_PRSRJCT1)	32	R	0000_0000h	43.5.120/ 1450
5B4	Parser Accept Count 2 (ENET_PRSACPT2)	32	R	0000_0000h	43.5.121/ 1451
5B8	Parser Reject Count 2 (ENET_PRSRJCT2)	32	R	0000_0000h	43.5.122/ 1451
604	Timer Global Status Register (ENET_TGSR)	32	R/W	0000_0000h	43.5.123/ 1452
608	Timer Control Status Register (ENET_TCSR0)	32	R/W	0000_0000h	43.5.124/ 1453
60C	Timer Compare Capture Register (ENET_TCCR0)	32	R/W	0000_0000h	43.5.125/ 1454
610	Timer Control Status Register (ENET_TCSR1)	32	R/W	0000_0000h	43.5.124/ 1453
614	Timer Compare Capture Register (ENET_TCCR1)	32	R/W	0000_0000h	43.5.125/ 1454

Table continues on the next page...

ENET memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
618	Timer Control Status Register (ENET_TCSR2)	32	R/W	0000_0000h	43.5.124/ 1453
61C	Timer Compare Capture Register (ENET_TCCR2)	32	R/W	0000_0000h	43.5.125/ 1454
620	Timer Control Status Register (ENET_TCSR3)	32	R/W	0000_0000h	43.5.124/ 1453
624	Timer Compare Capture Register (ENET_TCCR3)	32	R/W	0000_0000h	43.5.125/ 1454

43.5.1 Interrupt Event Register (ENET_EIR)

When an event occurs that sets a bit in EIR, an interrupt occurs if the corresponding bit in the interrupt mask register (EIMR) is also set. Writing a 1 to an EIR bit clears it; writing 0 has no effect. This register is cleared upon hardware reset.

NOTE

TxBD[INT] and RxBD[INT] must be set to 1 to allow setting the corresponding EIR register flags in enhanced mode, ENET_ECR[EN1588] = 1. Legacy mode does not require these flags to be enabled.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	PLR	WAKEUP	TS_AVAIL
W		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS_TIMER	RXFLUSH_2	RXFLUSH_1	RXFLUSH_0		PARSERR	PARSRF		TXF2	TXB2	RXF2	RXB2	TXF1	TXB1	RXF1	RXB1
W	w1c	w1c	w1c	w1c	0	w1c	w1c	0	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_EIR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 BABR	Babbling Receive Error Indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
2 BABT	Babbling Transmit Error Indicates the transmitted frame length exceeds RCR[MAX_FL] bytes. Usually this condition is caused when a frame that is too long is placed into the transmit data buffer(s). Truncation does not occur.
3 GRA	Graceful Stop Complete This interrupt is asserted after the transmitter is put into a pause state after completion of the frame currently being transmitted. See Graceful Transmit Stop (GTS) for conditions that lead to graceful stop. **NOTE:** The GRA interrupt is asserted only when the TX transitions into the stopped state. If this bit is cleared by writing 1 and the TX is still stopped, the bit is not set again.
4 TXF	Transmit Frame Interrupt Indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.
5 TXB	Transmit Buffer Interrupt Indicates a transmit buffer descriptor has been updated.
6 RXF	Receive Frame Interrupt Indicates a frame has been received and the last corresponding buffer descriptor has been updated.
7 RXB	Receive Buffer Interrupt Indicates a receive buffer descriptor is not the last in the frame has been updated.
8 MII	MII Interrupt. Indicates that the MII has completed the data transfer requested.
9 EBERR	Ethernet Bus Error Indicates a system bus error occurred when a uDMA transaction is underway. When this bit is set, ECR[ETHEREN] is cleared, halting frame processing by the MAC. When this occurs, software must ensure proper actions, possibly resetting the system, to resume normal operation.

Table continues on the next page...

ENET_EIR field descriptions (continued)

Field	Description
10 LC	Late Collision Indicates a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.
11 RL	Collision Retry Limit Indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half-duplex mode.
12 UN	Transmit FIFO Underrun Indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
13 PLR	Payload Receive Error Indicates a frame was received with a payload length error. See Frame Length/Type Verification: Payload Length Check for more information.
14 WAKEUP	Node Wakeup Request Indication Read-only status bit to indicate that a magic packet has been detected. Will act only if ECR[MAGICEN] is set.
15 TS_AVAIL	Transmit Timestamp Available Indicates that the timestamp of the last transmitted timing frame is available in the ATSTMP register.
16 TS_TIMER	Timestamp Timer The adjustable timer reached the period event. A period event interrupt can be generated if ATCR[PEREN] is set and the timer wraps according to the periodic setting in the ATPER register. Set the timer period value before setting ATCR[PEREN].
17 RXFLUSH_2	RX DMA Ring 2 flush indication. This ring's RX frame has been flushed due to either RDAR2[RDAR] or RxBD[E] being clear and only if QOS[RXFLUSH_2] is enabled.
18 RXFLUSH_1	RX DMA Ring 1 flush indication. This ring's RX frame has been flushed due to either RDAR1[RDAR] or RxBD[E] being clear and only if QOS[RXFLUSH_1] is enabled.
19 RXFLUSH_0	RX DMA Ring 0 flush indication. . This ring's RX frame has been flushed due to either RDAR[RDAR] or RxBD[E] being clear and only if QOS[RXFLUSH_0] is enabled.
20 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
21 PARSERR	Receive parser error or all entries of the match table checked without any match.
22 PARSRF	Receive frame rejected due to the match with the table entry with MCONFIG[RF] = 1.
23 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
24 TXF2	Transmit frame interrupt, class 2 This bit indicates that a frame has been transmitted and the last corresponding buffer descriptor has been updated (ring/class 2).
25 TXB2	Transmit buffer interrupt, class 2 This field indicates that a transmit buffer descriptor has been updated (ring/class 2).

Table continues on the next page...

ENET_EIR field descriptions (continued)

Field	Description
26 RXF2	Receive frame interrupt, class 2 This field indicates that a frame has been received and the last corresponding buffer descriptor has been updated (ring/class 2).
27 RXB2	Receive buffer interrupt, class 2 This field indicates that a receive buffer descriptor, that not the last in the frame, has been updated (ring/class 2).
28 TXF1	Transmit frame interrupt, class 1 This bit indicates that a frame has been transmitted and the last corresponding buffer descriptor has been updated (ring/class 1).
29 TXB1	Transmit buffer interrupt, class 1 This field indicates that a transmit buffer descriptor has been updated (ring/class 1).
30 RXF1	Receive frame interrupt, class 1 This field indicates that a frame has been received and the last corresponding buffer descriptor has been updated (ring/class 1).
31 RXB1	Receive buffer interrupt, class 1 This field indicates that a receive buffer descriptor, that not the last in the frame, has been updated (ring/class 1).

43.5.2 Interrupt Mask Register (ENET_EIMR)

EIMR controls which interrupt events are allowed to generate actual interrupts. A hardware reset clears this register. If the corresponding bits in the EIR and EIMR registers are set, an interrupt is generated. The interrupt signal remains asserted until a 1 is written to the EIR field (write 1 to clear) or a 0 is written to the EIMR field.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R		BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	PLR	WAKEUP	TS_AVAIL
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TS_TIMER	RXFLUSH_2	RXFLUSH_1	RXFLUSH_0		PARSERR	PARSRF		TXF2	TXB2	RXF2	RXB2	TXF1	TXB1	RXF1	RXB1
W					0			0								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_EIMR field descriptions

Field	Description
0 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
1 BABR	BABR Interrupt Mask Corresponds to interrupt source EIR[BABR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR BABR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
2 BABT	BABT Interrupt Mask Corresponds to interrupt source EIR[BABT] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR BABT field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
3 GRA	GRA Interrupt Mask Corresponds to interrupt source EIR[GRA] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR GRA field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
4 TXF	TXF Interrupt Mask Corresponds to interrupt source EIR[TXF] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
5 TXB	TXB Interrupt Mask Corresponds to interrupt source EIR[TXB] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked.
6 RXF	RXF Interrupt Mask Corresponds to interrupt source EIR[RXF] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.

Table continues on the next page...

ENET_EIMR field descriptions (continued)

Field	Description
7 RXB	RXB Interrupt Mask Corresponds to interrupt source EIR[RXB] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RXB field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
8 MII	MII Interrupt Mask Corresponds to interrupt source EIR[MII] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR MII field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
9 EBERR	EBERR Interrupt Mask Corresponds to interrupt source EIR[EBERR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR EBERR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
10 LC	LC Interrupt Mask Corresponds to interrupt source EIR[LC] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR LC field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
11 RL	RL Interrupt Mask Corresponds to interrupt source EIR[RL] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RL field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
12 UN	UN Interrupt Mask Corresponds to interrupt source EIR[UN] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR UN field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
13 PLR	PLR Interrupt Mask Corresponds to interrupt source EIR[PLR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR PLR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
14 WAKEUP	WAKEUP Interrupt Mask Corresponds to interrupt source EIR[WAKEUP] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR WAKEUP field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
15 TS_AVAIL	TS_AVAIL Interrupt Mask Corresponds to interrupt source EIR[TS_AVAIL] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting

Table continues on the next page...

ENET_EIMR field descriptions (continued)

Field	Description
	source. The corresponding EIR TS_AVAIL field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
16 TS_TIMER	TS_TIMER Interrupt Mask Corresponds to interrupt source EIR[TS_TIMER] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TS_TIMER field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
17 RXFLUSH_2	Corresponds to interrupt source EIR[RXFLUSH_2] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXFLUSH_2] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
18 RXFLUSH_1	Corresponds to interrupt source EIR[RXFLUSH_1] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXFLUSH_1] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
19 RXFLUSH_0	Corresponds to interrupt source EIR[RXFLUSH_0] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXFLUSH_0] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
20 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
21 PARSERR	Interrupt mask bit corresponding to EIR[PARSERR]. This field determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[PARSERR] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
22 PARSRF	Interrupt mask bit corresponding to EIR[PARSRF]. This field determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[PARSRF] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
23 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
24 TXF2	Transmit frame interrupt, class 2 Corresponds to interrupt source EIR[TXF2] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[TXF2] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
25 TXB2	Transmit buffer interrupt, class 2 Corresponds to interrupt source EIR[TXB2] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[TXB2] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
26 RXF2	Receive frame interrupt, class 2 Corresponds to interrupt source EIR[RXF2] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXF2] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
27 RXB2	Receive buffer interrupt, class 2

Table continues on the next page...

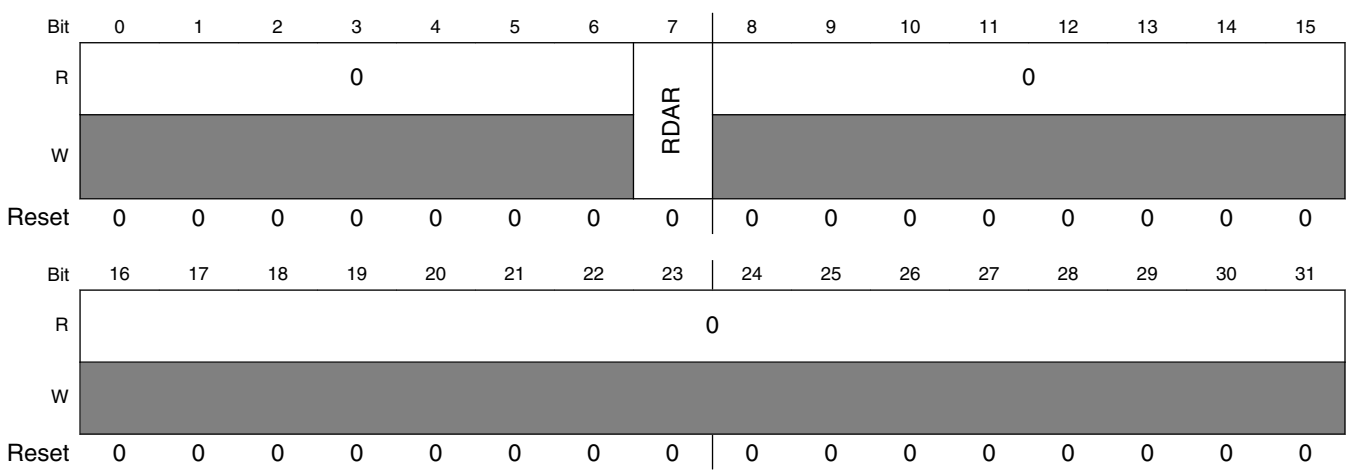
ENET_EIMR field descriptions (continued)

Field	Description
	Corresponds to interrupt source EIR[RXB2] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXB2] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
28 TXF1	Transmit frame interrupt, class 1 Corresponds to interrupt source EIR[TXF1] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[TXF1] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
29 TXB1	Transmit buffer interrupt, class 1 Corresponds to interrupt source EIR[TXB1] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[TXB1] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
30 RXF1	Receive frame interrupt, class 1 Corresponds to interrupt source EIR[RXF1] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXF1] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.
31 RXB1	Receive buffer interrupt, class 1 Corresponds to interrupt source EIR[RXB1] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. EIR[RXB1] reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.

43.5.3 Receive Descriptor Active Register - Ring 0 (ENET_RDAR)

RDAR is a command register, written by the user, to indicate that the receive descriptor ring 0 has been updated, that is, that the driver produced empty receive buffers with the empty bit set.

Address: 0h base + 10h offset = 10h



ENET_RDAR field descriptions

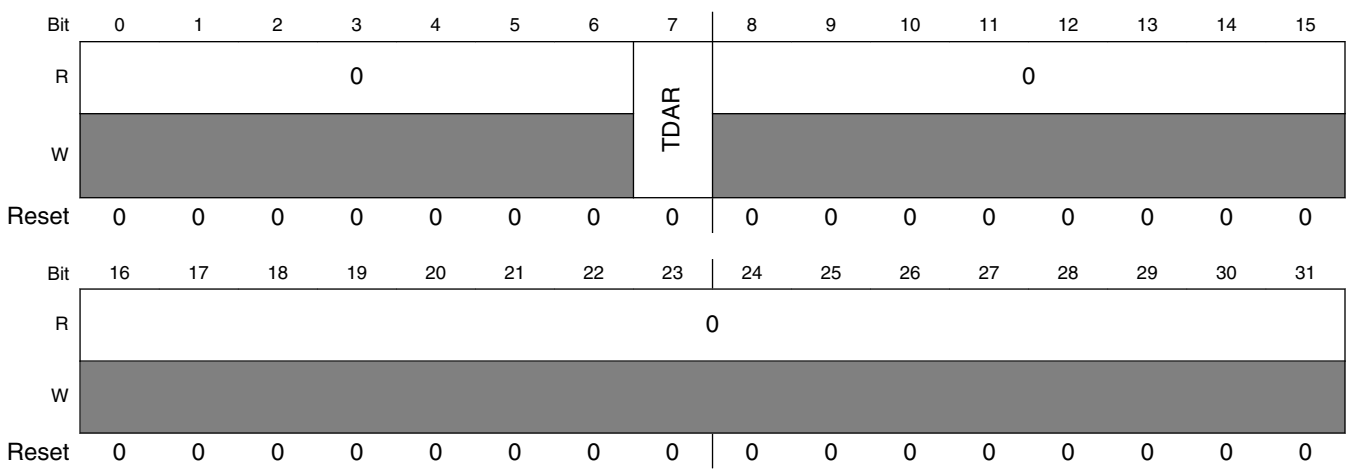
Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 RDAR	Receive Descriptor Active Always set to 1 when this register is written, regardless of the value written. This field is cleared by the MAC device when no additional empty descriptors remain in the receive ring. It is also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.4 Transmit Descriptor Active Register - Ring 0 (ENET_TDAR)

The TDAR is a command register that the user writes to indicate that the transmit descriptor ring 0 has been updated, that is, that transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor.

The TDAR register is cleared at reset, when ECR[ETHEREN] transitions from set to cleared, or when ECR[RESET] is set.

Address: 0h base + 14h offset = 14h



ENET_TDAR field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TDAR	Transmit Descriptor Active Always set to 1 when this register is written, regardless of the value written. This bit is cleared by the MAC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.

Table continues on the next page...

ENET_TDAR field descriptions (continued)

Field	Description
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.5 Ethernet Control Register (ENET_ECR)

ECR is a read/write user register, though hardware may also alter fields in this register. It controls many of the high level features of the Ethernet MAC, including legacy FEC support through the EN1588 field.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved														Reserved	
Reset	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved				SVLANDBL	VLANUSE2ND	SVLANEN	DBSWP	Reserved	DBGEN	0	EN1588	SLEEP	MAGICEN	ETHEREN	RESET
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ECR field descriptions

Field	Description
0–13 Reserved	This field is reserved. Always write 01110000000000b to this field.
14–19 Reserved	This field is reserved. Always write 0 to this field.
20 SVLANDBL	S-VLAN double tag If enabled, S-VLAN detection requires a double-tagged frame to define a frame as being a VLAN frame. The following rules apply: • If the first tag is the S-VLAN type, it must be followed by a second tag with the C-VLAN type to declare the frame as VLAN frame. • If the first tag is the S-VLAN type but no 2nd tag follows, it is not considered a VLAN frame but instead treated as an untagged frame. • If the first tag is the C-VLAN type, it is considered a VLAN frame as normal. **NOTE:** VLANUSE2ND can be used to determine from which tag the data should be extracted. This applies only if SVLAN_EN = 1, ignored otherwise.

Table continues on the next page...

ENET_ECR field descriptions (continued)

Field	Description
21 VLANUSE2ND	VLAN use second tag 0 Always extract data from the first VLAN tag if it exists. 1 When a double-tagged frame is detected, the data of the second tag is extracted for further processing. A double-tagged frame is defined as: • The first tag can be a C-VLAN or a S-VLAN (if SVLAN_ENA = 1) • The second tag must be a C-VLAN
22 SVLANEN	S-VLAN enable Enable additional detection of S-VLAN tag according to IEEE802.1Q. 0 Only the EtherType 0x8100 will be considered for VLAN detection. 1 The EtherType 0x88a8 will be considered in addition to 0x8100 (C-VLAN) to identify a VLAN frame in receive. When a VLAN frame is identified, the two bytes following the VLAN type are extracted and used by the classification match comparators, RCMRn.
23 DBSWP	Descriptor Byte Swapping Enable Swaps the byte locations of the buffer descriptors. **NOTE:** This field resets to 0 and must not be changed. 0 The buffer descriptor bytes are not swapped to support big-endian devices. 1 The buffer descriptor bytes are swapped to support little-endian devices.
24 Reserved	This field is reserved. Always write 0 to this field.
25 DBGEN	Debug Enable Enables the MAC to enter hardware freeze mode when the device enters debug mode. 0 MAC continues operation in debug mode. 1 MAC enters hardware freeze mode when the processor is in debug mode.
26 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
27 EN1588	EN1588 Enable Enables enhanced functionality of the MAC. 0 Legacy FEC buffer descriptors and functions enabled. 1 Enhanced frame time-stamping functions enabled.
28 SLEEP	Sleep Mode Enable 0 Normal operating mode. 1 Sleep mode.
29 MAGICEN	Magic Packet Detection Enable Enables/disables magic packet detection. **NOTE:** MAGICEN is relevant only if the SLEEP field is set. If MAGICEN is set, changing the SLEEP field enables/disables sleep mode and magic packet detection. 0 Magic detection logic disabled. 1 The MAC core detects magic packets and asserts EIR[WAKEUP] when a frame is detected.

Table continues on the next page...

ENET_ECR field descriptions (continued)

Field	Description
30 ETHEREN	Ethernet Enable Enables/disables the Ethernet MAC. When the MAC is disabled, the buffer descriptors for an aborted transmit frame are not updated. The uDMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware clears this field under the following conditions: • RESET is set by software • An error condition causes the EBERR field to set. **NOTE:** • ETHEREN must be set at the very last step during ENET configuration/setup/initialization, only *after* all other ENET-related registers have been configured. • If ETHEREN is cleared to 0 by software then next time ETHEREN is set, the EIR interrupts must cleared to 0 due to previous pending interrupts. 0 Reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. 1 MAC is enabled, and reception and transmission are possible.
31 RESET	Ethernet MAC Reset When this field is set, it clears the ETHEREN field.

43.5.6 MII Management Frame Register (ENET_MMFR)

Writing to MMFR triggers a management frame transaction to the PHY device unless MSCR is programmed to zero.

If MSCR is changed from zero to non-zero during a write to MMFR, an MII frame is generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

If the MMFR register is written while frame generation is in progress, the frame contents are altered. Software must use the EIR[MII] interrupt indication to avoid writing to the MMFR register while frame generation is in progress.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ST		OP		PA					RA					TA		DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_MMFR field descriptions

Field	Description
0–1 ST	Start Of Frame Delimiter See Table 43-41 (Clause 22) or Table 43-43 (Clause 45) for correct value.

Table continues on the next page...

ENET_MMFR field descriptions (continued)

Field	Description
2–3 OP	Operation Code See Table 43-41 (Clause 22) or Table 43-43 (Clause 45) for correct value.
4–8 PA	PHY Address See Table 43-41 (Clause 22) or Table 43-43 (Clause 45) for correct value.
9–13 RA	Register Address See Table 43-41 (Clause 22) or Table 43-43 (Clause 45) for correct value.
14–15 TA	Turn Around This field must be programmed to 10 to generate a valid MII management frame.
16–31 DATA	Management Frame Data This is the field for data to be written to or read from the PHY register.

43.5.7 MII Speed Control Register (ENET_MSCR)

MSCR provides control of the MII clock (MDC pin) frequency and allows a preamble drop on the MII management frame.

The MII_SPEED field must be programmed with a value to provide an MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value to source a read or write management frame. After the management frame is complete, the MSCR register may optionally be cleared to turn off MDC. The MDC signal generated has a 50% duty cycle except when MII_SPEED changes during operation. This change takes effect following a rising or falling edge of MDC.

For example, if the internal module clock (i.e., IPS bus clock) is 25 MHz, programming MII_SPEED to 0x4 results in an MDC as given in the following equation:

MII clock frequency = 25 MHz / ((4 + 1) x 2) = 2.5 MHz

The following table shows the optimum values for MII_SPEED as a function of IPS bus clock frequency.

Table 43-3. Programming Examples for MSCR

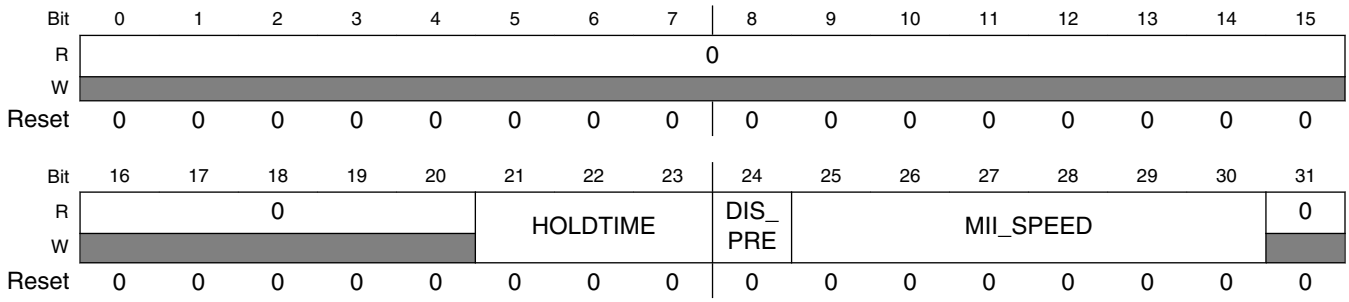
Internal module clock frequency	MSCR [MII_SPEED]	MDC frequency
25 MHz	0x4	2.50 MHz
33 MHz	0x6	2.36 MHz
40 MHz	0x7	2.50 MHz
50 MHz	0x9	2.50 MHz

Table continues on the next page...

Table 43-3. Programming Examples for MSCR (continued)

Internal module clock frequency	MSCR [MII_SPEED]	MDC frequency
66 MHz	0xD	2.36 MHz

Address: 0h base + 44h offset = 44h



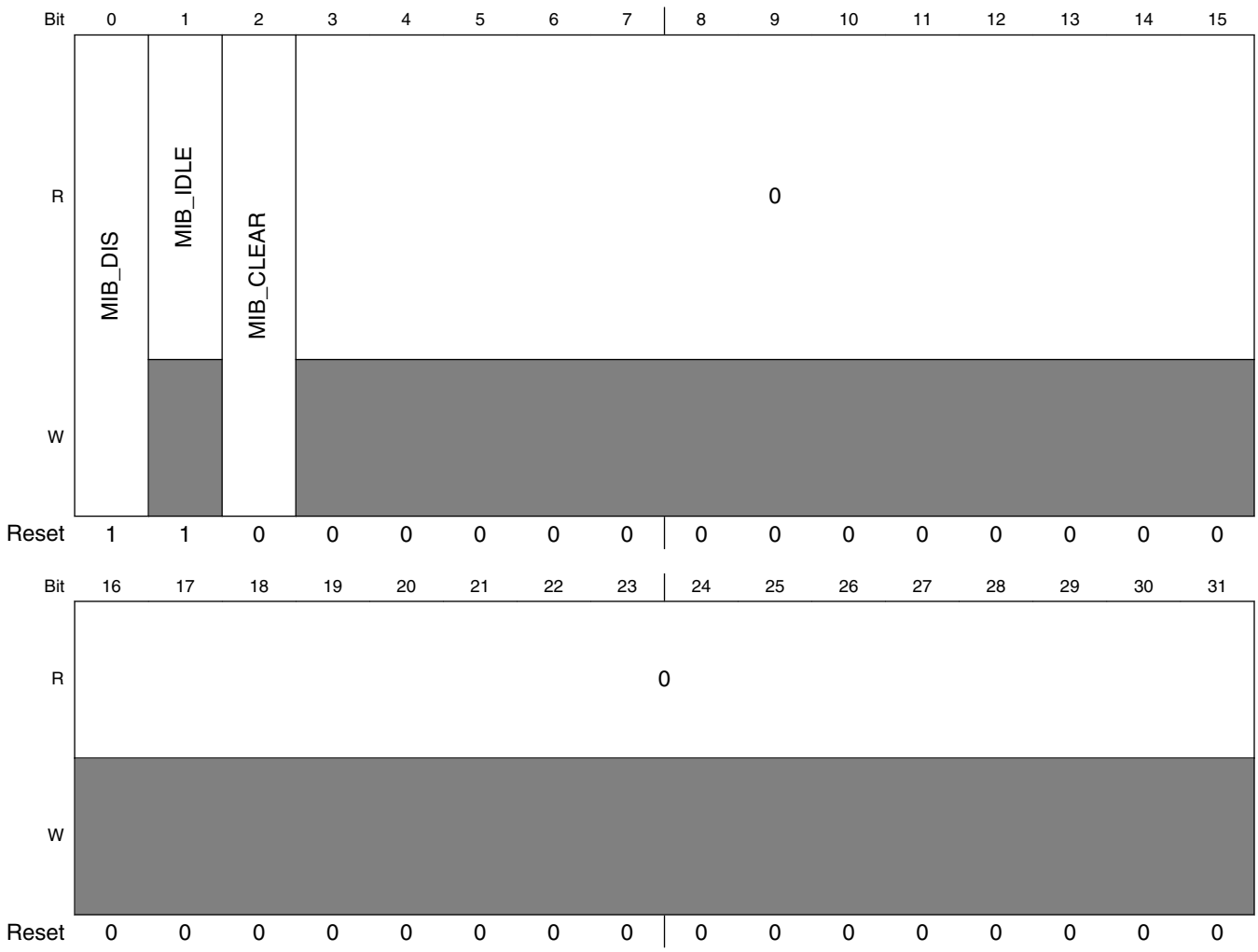
ENET_MSCR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–23 HOLDTIME	Hold time On MDIO Output IEEE802.3 clause 22 defines a minimum of 10 ns for the hold time on the MDIO output. Depending on the host bus frequency, the setting may need to be increased. 000 1 internal module clock cycle 001 2 internal module clock cycles 010 3 internal module clock cycles 111 8 internal module clock cycles
24 DIS_PRE	Disable Preamble Enables/disables prepending a preamble to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY devices do not require it. 0 Preamble enabled. 1 Preamble (32 ones) is not prepended to the MII management frame.
25–30 MII_SPEED	MII Speed Controls the frequency of the MII management interface clock (MDC) relative to the internal module clock. A value of 0 in this field turns off MDC and leaves it in low voltage state. Any non-zero value results in the MDC frequency of: 1/((MII_SPEED + 1) x 2) of the internal module clock frequency
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.8 MIB Control Register (ENET_MIBC)

MIBC is a read/write register controlling and observing the state of the MIB block. Access this register to disable the MIB block operation or clear the MIB counters. The MIB_DIS field resets to 1.

Address: 0h base + 64h offset = 64h



ENET_MIBC field descriptions

Field	Description
0 MIB_DIS	Disable MIB Logic If this control field is set, 0 MIB logic is enabled. 1 MIB logic is disabled. The MIB logic halts and does not update any MIB counters.

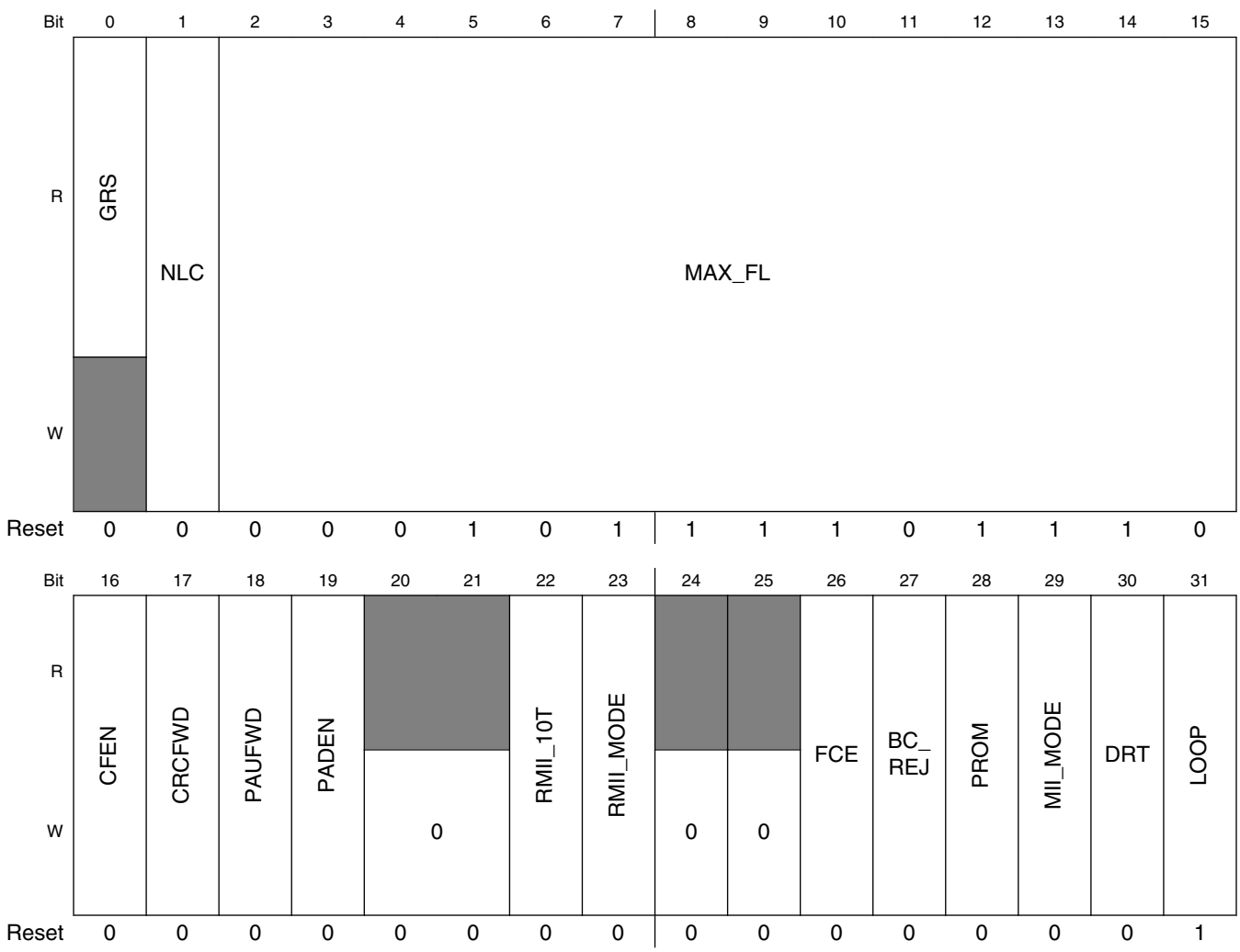
Table continues on the next page...

ENET_MIBC field descriptions (continued)

Field	Description
1 MIB_IDLE	MIB Idle 0 The MIB block is updating MIB counters. 1 The MIB block is not currently updating any MIB counters.
2 MIB_CLEAR	MIB Clear **NOTE:** This field is not self-clearing. To clear the MIB counters set and then clear this field. 0 See note above. 1 All statistics counters are reset to 0.
3–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.9 Receive Control Register (ENET_RCR)

Address: 0h base + 84h offset = 84h



ENET_RCR field descriptions

Field	Description
0 GRS	Graceful Receive Stopped Read-only status indicating that the MAC receive datapath is stopped.
1 NLC	Payload Length Check Disable Enables/disables a payload length check. 0 The payload length check is disabled. 1 The core checks the frame's payload length with the frame length/type field. Errors are indicated in the EIR[PLC] field.
2–15 MAX_FL	Maximum Frame Length Resets to decimal 1518. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL cause the BABT interrupt to occur. Receive frames longer than MAX_FL cause the BABR interrupt to occur and set the LG field in the end of frame receive buffer descriptor. The recommended default value to be programmed is 1518 or 1522 if VLAN tags are supported.
16 CFEN	MAC Control Frame Enable Enables/disables the MAC control frame. 0 MAC control frames with any opcode other than 0x0001 (pause frame) are accepted and forwarded to the client interface. 1 MAC control frames with any opcode other than 0x0001 (pause frame) are silently discarded.
17 CRCFWD	Terminate/Forward Received CRC Specifies whether the CRC field of received frames is transmitted or stripped. **NOTE:** If padding function is enabled (PADEN = 1), CRCFWD is ignored and the CRC field is checked and always terminated and removed. 0 The CRC field of received frames is transmitted to the user application. 1 The CRC field is stripped from the frame.
18 PAUFWD	Terminate/Forward Pause Frames Specifies whether pause frames are terminated or forwarded. 0 Pause frames are terminated and discarded in the MAC. 1 Pause frames are forwarded to the user application.
19 PADEN	Enable Frame Padding Remove On Receive Specifies whether the MAC removes padding from received frames. 0 No padding is removed on receive by the MAC. 1 Padding is removed from received frames.
20–21 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
22 RMII_10T	Enables 10-Mbit/s mode of the RMII . 0 100-Mbit/s operation. 1 10-Mbit/s operation.

Table continues on the next page...

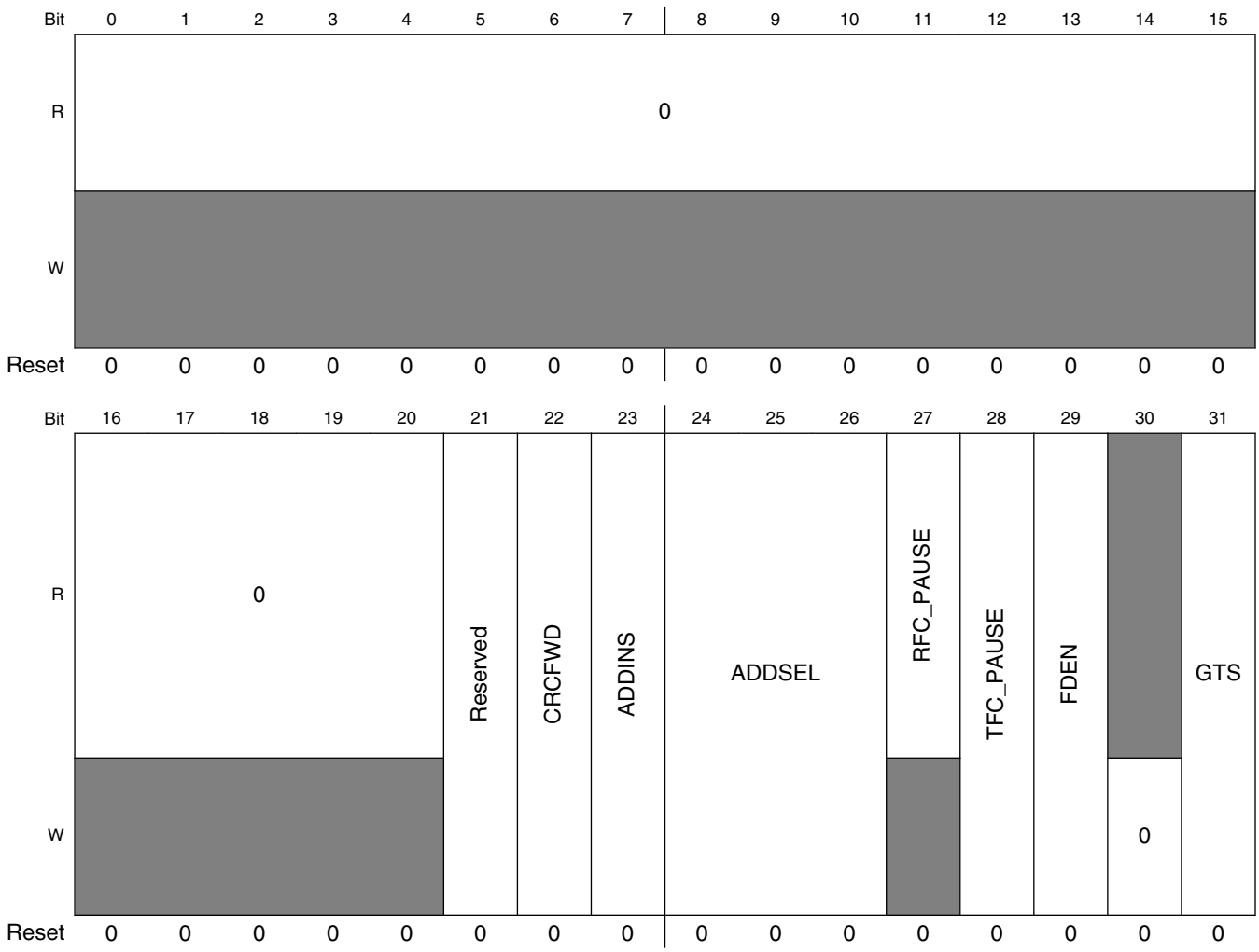
ENET_RCR field descriptions (continued)

Field	Description
23 RMII_MODE	RMII Mode Enable Specifies whether the MAC is configured for MII mode or RMII operation . 0 MAC configured for MII mode. 1 MAC configured for RMII operation.
24 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
25 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
26 FCE	Flow Control Enable If set, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter stops transmitting data frames for a given duration.
27 BC_REJ	Broadcast Frame Reject If set, frames with destination address (DA) equal to 0xFFFF_FFFF_FFFF are rejected unless the PROM field is set. If BC_REJ and PROM are set, frames with broadcast DA are accepted and the MISS (M) is set in the receive buffer descriptor.
28 PROM	Promiscuous Mode All frames are accepted regardless of address matching. 0 Disabled. 1 Enabled.
29 MII_MODE	Media Independent Interface Mode This field must always be set. 0 Reserved. 1 MII or RMII mode, as indicated by the RMII_MODE field.
30 DRT	Disable Receive On Transmit 0 Receive path operates independently of transmit (i.e., full-duplex mode). Can also be used to monitor transmit activity in half-duplex mode. 1 Disable reception of frames while transmitting. (Normally used for half-duplex mode.)
31 LOOP	Internal Loopback This is an MII internal loopback, therefore MII_MODE must be written to 1 and RMII_MODE must be written to 0. 0 Loopback disabled. 1 Transmitted frames are looped back internal to the device and transmit MII output signals are not asserted. DRT must be cleared.

43.5.10 Transmit Control Register (ENET_TCR)

TCR is read/write and configures the transmit block. This register is cleared at system reset. FDEN can only be modified when ECR[ETHEREN] is cleared.

Address: 0h base + C4h offset = C4h



ENET_TCR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This field is read/write and must be set to 0.
22 CRCFWD	Forward Frame From Application With CRC

Table continues on the next page...

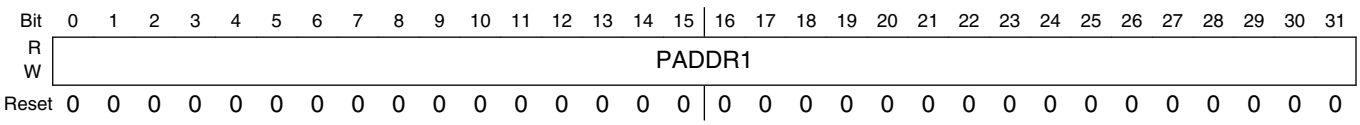
ENET_TCR field descriptions (continued)

Field	Description
	0 TxBD[TC] controls whether the frame has a CRC from the application. 1 The transmitter does not append any CRC to transmitted frames, as it is expecting a frame with CRC from the application.
23 ADDINS	Set MAC Address On Transmit 0 The source MAC address is not modified by the MAC. 1 The MAC overwrites the source MAC address with the programmed MAC address according to ADDSEL.
24–26 ADDSEL	Source MAC Address Select On Transmit If ADDINS is set, indicates the MAC address that overwrites the source MAC address. 000 Node MAC address programmed on PADDR1/2 registers. 100 Reserved. 101 Reserved. 110 Reserved.
27 RFC_PAUSE	Receive Frame Control Pause This status field is set when a full-duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This field automatically clears when the pause duration is complete.
28 TFC_PAUSE	Transmit Frame Control Pause Pauses frame transmission. When this field is set, EIR[GRA] is set. With transmission of data frames stopped, the MAC transmits a MAC control PAUSE frame. Next, the MAC clears TFC_PAUSE and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC control PAUSE frame. 0 No PAUSE frame transmitted. 1 The MAC stops transmission of data frames after the current transmission is complete.
29 FDEN	Full-Duplex Enable If this field is set, frames transmit independent of carrier sense and collision inputs. Only modify this bit when ECR[ETHEREN] is cleared.
30 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
31 GTS	Graceful Transmit Stop When this field is set, MAC stops transmission after any frame currently transmitted is complete and EIR[GRA] is set. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission finishes, clear GTS to restart. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS is set, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this, clear ECR[ETHEREN] following the GRA interrupt.

43.5.11 Physical Address Lower Register (ENET_PALR)

PALR contains the lower 32 bits (bytes 0, 1, 2, 3) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the six-byte source address field when transmitting PAUSE frames.

Address: 0h base + E4h offset = E4h



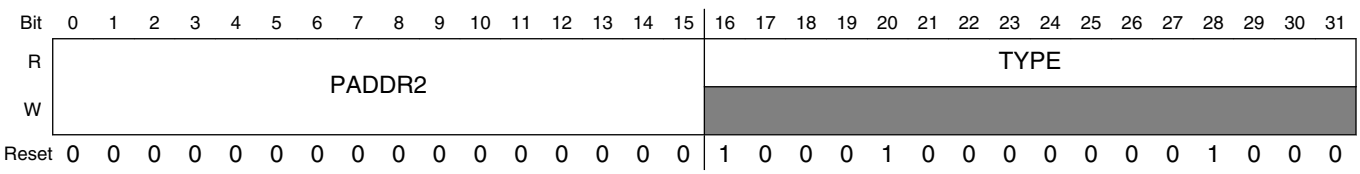
ENET_PALR field descriptions

Field	Description
0–31 PADDR1	Pause Address Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

43.5.12 Physical Address Upper Register (ENET_PAUR)

PAUR contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the six-byte source address field when transmitting PAUSE frames. Bits 15:0 of PAUR contain a constant type field (0x8808) for transmission of PAUSE frames.

Address: 0h base + E8h offset = E8h



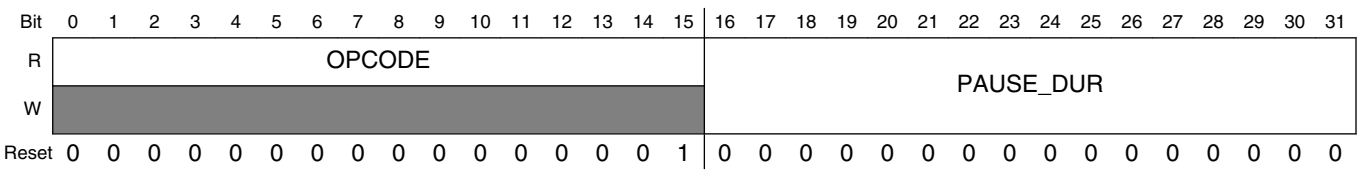
ENET_PAUR field descriptions

Field	Description
0–15 PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
16–31 TYPE	Type Field In PAUSE Frames These fields have a constant value of 0x8808.

43.5.13 Opcode/Pause Duration Register (ENET_OPD)

OPD is read/write accessible. This register contains the 16-bit opcode and 16-bit pause duration fields used in transmission of a PAUSE frame. The opcode field is a constant value, 0x0001. When another node detects a PAUSE frame, that node pauses transmission for the duration specified in the pause duration field. The lower 16 bits of this register are not reset and you must initialize it.

Address: 0h base + ECh offset = ECh



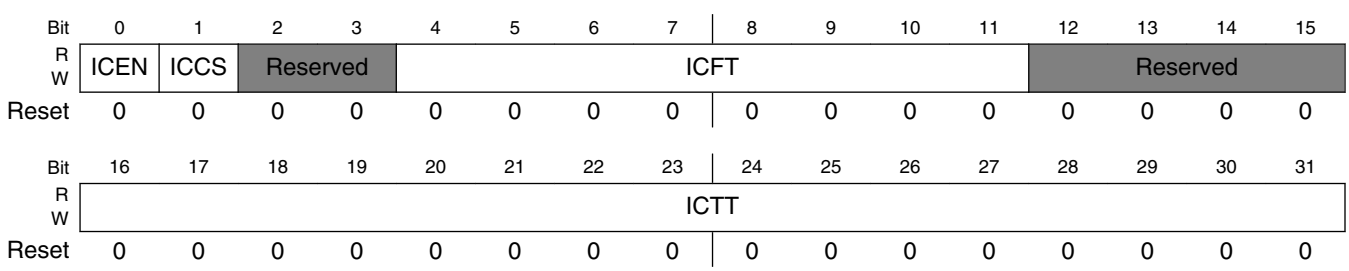
ENET_OPD field descriptions

Field	Description
0–15 OPCODE	Opcode Field In PAUSE Frames These fields have a constant value of 0x0001.
16–31 PAUSE_DUR	Pause Duration Pause duration field used in PAUSE frames.

43.5.14 Transmit Interrupt Coalescing Register (ENET_TXIC*n*)

See Interrupt coalescence for more information.

Address: 0h base + F0h offset + (4d × i), where i=0d to 2d



ENET_TXIC*n* field descriptions

Field	Description
0 ICEN	Interrupt Coalescing Enable

Table continues on the next page...

ENET_TXIC*n* field descriptions (continued)

Field	Description
	0 Disable Interrupt coalescing. 1 Enable Interrupt coalescing.
1 ICCS	Interrupt Coalescing Timer Clock Source Select 0 Use MII/GMII TX clocks. 1 Use ENET system clock.
2–3 Reserved	This field must be set to 0. This field is reserved.
4–11 ICFT	Interrupt coalescing frame count threshold This value determines the number of frames needed to be transmitted for raising an interrupt. Frame counter restarts after reaching this threshold value or after the expiring of the coalescing timer. Must be greater than zero to avoid unpredictable behavior.
12–15 Reserved	This field must be set to 0. This field is reserved.
16–31 ICTT	Interrupt coalescing timer threshold Interrupt coalescing timer threshold in units of 64 clock periods. This value determines the maximum amount of time after transmitting a frame before raising an interrupt. The threshold timer is disabled after expiring or number of frame transmission defined by ICFT and starts again upon transmission of the next first frame. Must be greater than zero to avoid unpredictable behavior.

43.5.15 Receive Interrupt Coalescing Register (ENET_RXIC*n*)

See Interrupt coalescence for more information.

Address: 0h base + 100h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ICEN	ICCS	Reserved		ICFT								Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ICTT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RXIC*n* field descriptions

Field	Description
0 ICEN	Interrupt Coalescing Enable 0 Disable Interrupt coalescing. 1 Enable Interrupt coalescing.

Table continues on the next page...

ENET_RXIC*n* field descriptions (continued)

Field	Description
1 ICCS	Interrupt Coalescing Timer Clock Source Select 0 Use MII/GMII TX clocks. 1 Use ENET system clock.
2–3 Reserved	This field must be set to 0. This field is reserved.
4–11 ICFT	Interrupt coalescing frame count threshold This value determines the number of frames needed to be received for raising an interrupt. Frame counter restarts after reaching this threshold value or after the expiring of the coalescing timer. Must be greater than zero to avoid unpredictable behavior.
12–15 Reserved	This field must be set to 0. This field is reserved.
16–31 ICTT	Interrupt coalescing timer threshold Interrupt coalescing timer threshold in units of 64 clock periods. This value determines the maximum amount of time after receiving a frame before raising an interrupt. The threshold timer is disabled after expiring or number of frame reception defined by ICFT and starts again upon reception of the next first frame. Must be greater than zero to avoid unpredictable behavior.

43.5.16 Descriptor Individual Upper Address Register (ENET_IAUR)

IAUR contains the upper 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the destination address (DA) field of receive frames with an individual DA. This register is not reset and you must initialize it.

Address: 0h base + 118h offset = 118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IAUR field descriptions

Field	Description
0–31 IADDR1	Contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

43.5.17 Descriptor Individual Lower Address Register (ENET_IALR)

IALR contains the lower 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the DA field of receive frames with an individual DA. This register is not reset and you must initialize it.

Address: 0h base + 11Ch offset = 11Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IADDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IALR field descriptions

Field	Description
0–31 IADDR2	Contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

43.5.18 Descriptor Group Upper Address Register (ENET_GAUR)

GAUR contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

Address: 0h base + 120h offset = 120h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_GAUR field descriptions

Field	Description
0–31 GADDR1	Contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

43.5.19 Descriptor Group Lower Address Register (ENET_GALR)

GALR contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

Address: 0h base + 124h offset = 124h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GADDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_GALR field descriptions

Field	Description
0–31 GADDR2	Contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

43.5.20 Transmit FIFO Watermark Register (ENET_TFWR)

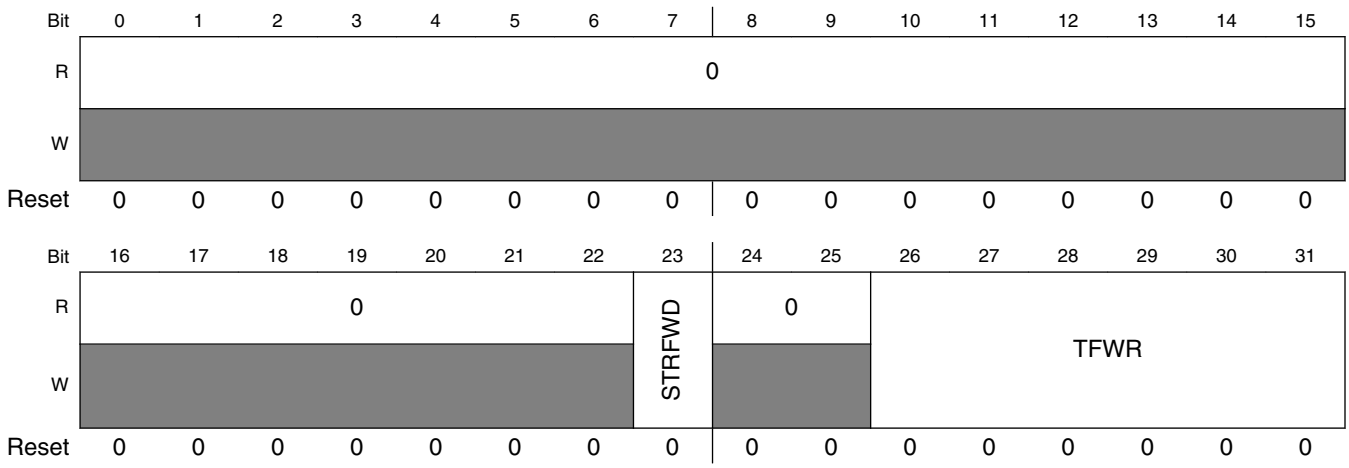
If TFWR[STRFWD] is cleared, TFWR[TFWR] controls the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows you to minimize transmit latency (TFWR = 00 or 01) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement, for example, worst-case bus access latency by the transmit data uDMA channel.

When the FIFO level reaches the value the TFWR field and when the STR_FWD is set to ‘0’, the MAC transmit control logic starts frame transmission even before the end-of-frame is available in the FIFO (cut-through operation).

If a complete frame has a size smaller than the threshold programmed with TFWR, the MAC also transmits the Frame to the line.

To enable store and forward on the Transmit path, set STR_FWD to ‘1’. In this case, the MAC starts to transmit data only when a complete frame is stored in the Transmit FIFO.

Address: 0h base + 144h offset = 144h



ENET_TFWR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 STRFWD	Store And Forward Enable 0 Reset. The transmission start threshold is programmed in TFWR[TFWR]. 1 Enabled.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 TFWR	Transmit FIFO Write If TFWR[STRFWD] is cleared, this field indicates the number of bytes, in steps of 64 bytes, written to the transmit FIFO before transmission of a frame begins. **NOTE:** If a frame with less than the threshold is written, it is still sent independently of this threshold setting. The threshold is relevant only if the frame is larger than the threshold given. 000000 64 bytes written. 000001 64 bytes written. 000010 128 bytes written. 000011 192 bytes written. ... ... 011111 1984 bytes written.

43.5.21 Receive Descriptor Ring 1 Start Register (ENET_RDSR1)

RDSR1 points to the beginning of circular receive buffer descriptor queue 1 in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation.

Address: 0h base + 160h offset = 160h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	R_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	R_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RDSR1 field descriptions

Field	Description
0–28 R_DES_START	Pointer to the beginning of the receive buffer descriptor queue 1.
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.22 Transmit Buffer Descriptor Ring 1 Start Register (ENET_TDSR1)

TDSR1 provides a pointer to the beginning of the circular transmit buffer descriptor queue 1 in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation.

Address: 0h base + 164h offset = 164h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	X_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	X_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TDSR1 field descriptions

Field	Description
0–28 X_DES_START	Pointer to the beginning of transmit buffer descriptor queue 1.

Table continues on the next page...

ENET_TDSR1 field descriptions (continued)

Field	Description
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.23 Maximum Receive Buffer Size Register - Ring 1 (ENET_MRBR1)

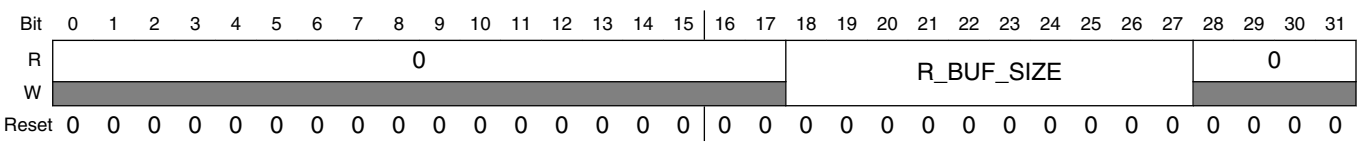
MRBR1 is a user-programmable register that dictates the maximum size of all ring-1 receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer.

- To allow one maximum size frame per buffer, MRBR1 must be set to RCR[MAX_FL] or larger.
- R_BUF_SIZE is concatentated with the four least-significant bits of this register and are used as the maximum receive buffer size.
- To properly align the buffer, MRBR1 must be evenly divisible by 64. To ensure this, set the lower two bits of R_BUF_ZIZE to zero. The lower four bits are already set to zero by the device.
- To minimize bus usage (descriptor fetches), set MRBR1 greater than or equal to 256 bytes.

NOTE

This register must be initialized before operation.

Address: 0h base + 168h offset = 168h



ENET_MRBR1 field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–27 R_BUF_SIZE	Receive buffer size in bytes. This value, concatenated with the four least-significant bits of this register (which are always zero), is the effective maximum receive buffer size.
28–31 Reserved	This field, which is always zero, is the four least-significant bits of the maximum receive buffer size. This field is reserved. This read-only field is reserved and always has the value 0.

43.5.24 Receive Descriptor Ring 2 Start Register (ENET_RDSR2)

RDSR points to the beginning of circular receive buffer descriptor queue 2 in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation

Address: 0h base + 16Ch offset = 16Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	R_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	R_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RDSR2 field descriptions

Field	Description
0–28 R_DES_START	Pointer to the beginning of receive buffer descriptor queue 2.
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.25 Transmit Buffer Descriptor Ring 2 Start Register (ENET_TDSR2)

TDSR2 provides a pointer to the beginning of circular transmit buffer descriptor queue 2 in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation

Address: 0h base + 170h offset = 170h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	X_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	X_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TDSR2 field descriptions

Field	Description
0–28 X_DES_START	Pointer to the beginning of transmit buffer descriptor queue 2.
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.26 Maximum Receive Buffer Size Register - Ring 2 (ENET_MRBR2)

MRBR2 is a user-programmable register that dictates the maximum size of all ring-2 receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer.

- To allow one maximum size frame per buffer, MRBR2 must be set to RCR[MAX_FL] or larger.
- R_BUF_SIZE is concatentated with the four least-significant bits of this register and are used as the maximum receive buffer size.
- To properly align the buffer, MRBR2 must be evenly divisible by 64. To ensure this, set the lower two bits of R_BUF_SIZE to zero. The lower four bits are already set to zero by the device.
- To minimize bus usage (descriptor fetches), set MRBR2 greater than or equal to 256 bytes.

NOTE

This register must be initialized prior to operation

Address: 0h base + 174h offset = 174h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																R_BUF_SIZE												0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_MRBR2 field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–27 R_BUF_SIZE	Receive buffer size in bytes. This value, concatenated with the four least-significant bits of this register (which are always zero), is the effective maximum receive buffer size.
28–31 Reserved	This field, which is always zero, is the four least-significant bits of the maximum receive buffer size. This field is reserved. This read-only field is reserved and always has the value 0.

43.5.27 Receive Descriptor Ring 0 Start Register (ENET_RDSR)

RDSR points to the beginning of the circular receive buffer descriptor queue in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation

Address: 0h base + 180h offset = 180h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	R_DES_START															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	R_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RDSR field descriptions

Field	Description
0–28 R_DES_START	Pointer to the beginning of the receive buffer descriptor queue. 0
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

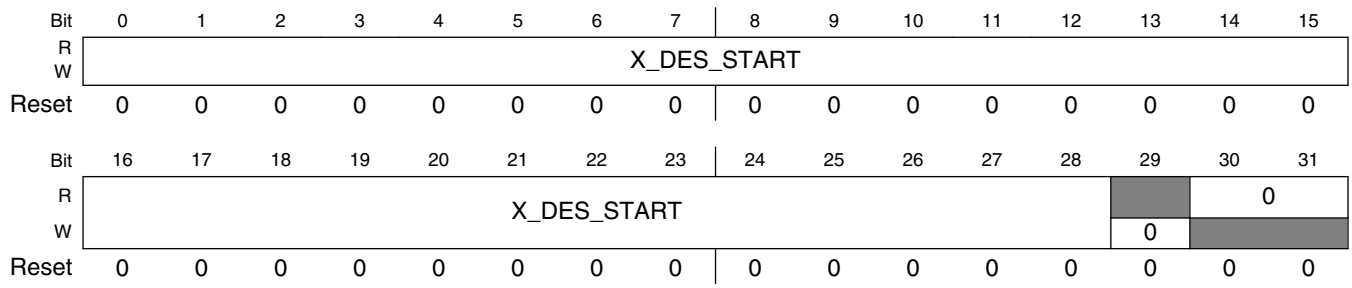
43.5.28 Transmit Buffer Descriptor Ring 0 Start Register (ENET_TDSR)

TDSR provides a pointer to the beginning of the circular transmit buffer descriptor queue 0 in external memory. This pointer must be 64-bit aligned (bits 29–31 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation.

Address: 0h base + 184h offset = 184h



ENET_TDSR field descriptions

Field	Description
0–28 X_DES_START	Pointer to the beginning of the transmit buffer descriptor queue.
29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.29 Maximum Receive Buffer Size Register - Ring 0 (ENET_MRBR)

The MRBR is a user-programmable register that dictates the maximum size of all ring-0 receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer.

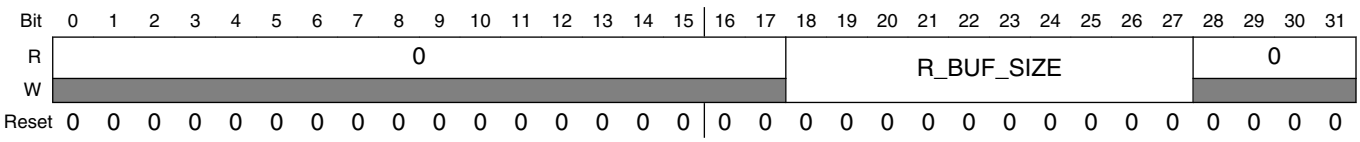
- R_BUF_SIZE is concatentated with the four least-significant bits of this register and are used as the maximum receive buffer size.
- To allow one maximum size frame per buffer, MRBR must be set to RCR[MAX_FL] or larger.

- To properly align the buffer, MRBR must be evenly divisible by 64. To ensure this, set the lower two bits of R_BUF_SIZE to zero. The lower four bits of this register are already set to zero by the device.
- To minimize bus usage (descriptor fetches), set MRBR greater than or equal to 256 bytes.

NOTE

This register must be initialized before operation.

Address: 0h base + 188h offset = 188h

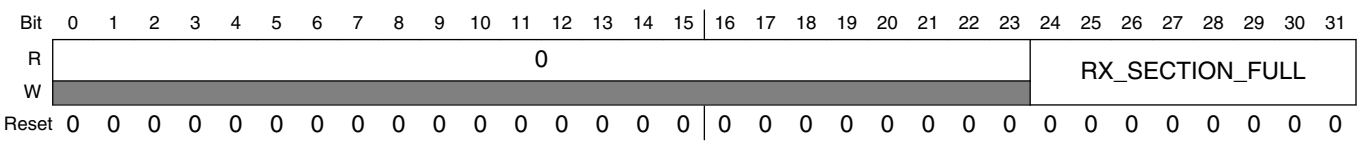


ENET_MRBR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–27 R_BUF_SIZE	Receive buffer size in bytes. This value, concatenated with the four least-significant bits of this register (which are always zero), is the effective maximum receive buffer size.
28–31 Reserved	This field, which is always zero, is the four least-significant bits of the maximum receive buffer size. This field is reserved. This read-only field is reserved and always has the value 0.

43.5.30 Receive FIFO Section Full Threshold (ENET_RSFL)

Address: 0h base + 190h offset = 190h

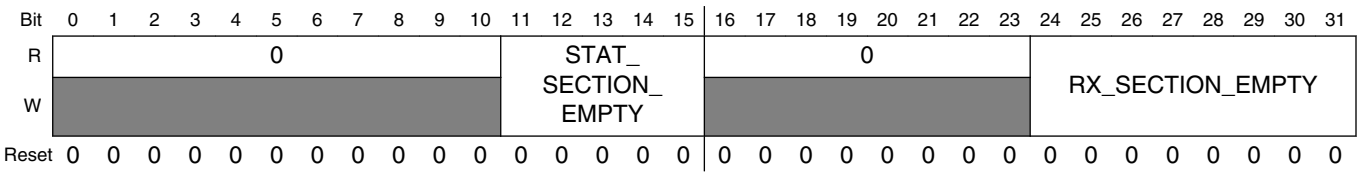


ENET_RSFL field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RX_SECTION_ FULL	Value Of Receive FIFO Section Full Threshold Value, in 64-bit words, of the receive FIFO section full threshold. Clear this field to enable store and forward on the RX FIFO. When programming a value greater than 0 (cut-through operation), it must be greater than RAEM[RX_ALMOST_EMPTY]. When the FIFO level reaches the value in this field, data is available in the Receive FIFO (cut-through operation).

43.5.31 Receive FIFO Section Empty Threshold (ENET_RSEM)

Address: 0h base + 194h offset = 194h

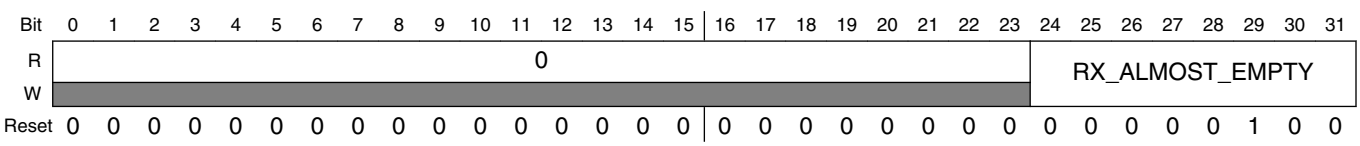


ENET_RSEM field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 STAT_ SECTION_ EMPTY	RX Status FIFO Section Empty Threshold Defines number of frames in the receive FIFO, independent of its size, that can be accepted. If the limit is reached, reception will continue normally, however a pause frame will be triggered to indicate a possible congestion to the remote device to avoid FIFO overflow. A value of 0 disables automatic pause frame generation
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RX_SECTION_ EMPTY	Value Of The Receive FIFO Section Empty Threshold Value, in 64-bit words, of the receive FIFO section empty threshold. When the FIFO has reached this level, a pause frame will be issued. A value of 0 disables automatic pause frame generation. When the FIFO level goes below the value programmed in this field, an XON pause frame is issued to indicate the FIFO congestion is cleared to the remote Ethernet client. **NOTE:** The section-empty threshold indications from both FIFOs are OR'ed to cause XOFF pause frame generation.

43.5.32 Receive FIFO Almost Empty Threshold (ENET_RAEM)

Address: 0h base + 198h offset = 198h



ENET_RAEM field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

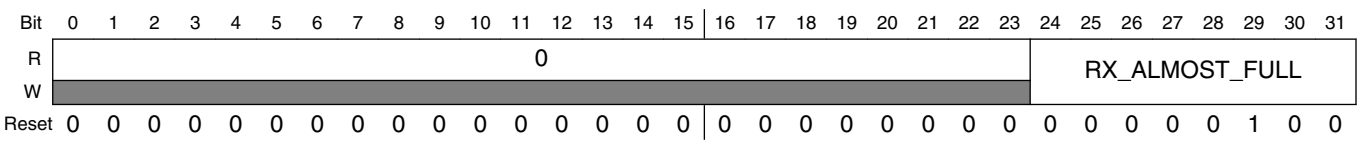
Table continues on the next page...

ENET_RAEM field descriptions (continued)

Field	Description
24–31 RX_ALMOST_ EMPTY	Value Of The Receive FIFO Almost Empty Threshold Value, in 64-bit words, of the receive FIFO almost empty threshold. When the FIFO level reaches the value programmed in this field and the end-of-frame has not been received for the frame yet, the core receive read control stops FIFO read (and subsequently stops transferring data to the MAC client application). It continues to deliver the frame, if again more data than the threshold or the end-of-frame is available in the FIFO. A minimum value of 4 should be set.

43.5.33 Receive FIFO Almost Full Threshold (ENET_RAFL)

Address: 0h base + 19Ch offset = 19Ch

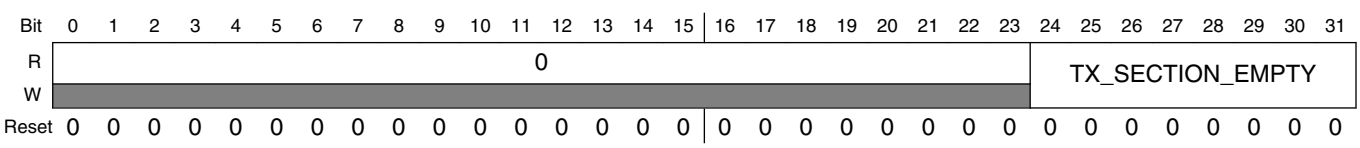


ENET_RAFL field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 RX_ALMOST_ FULL	Value Of The Receive FIFO Almost Full Threshold Value, in 64-bit words, of the receive FIFO almost full threshold. When the FIFO level comes close to the maximum, so that there is no more space for at least RX_ALMOST_FULL number of words, the MAC stops writing data in the FIFO and truncates the received frame to avoid FIFO overflow. The corresponding error status will be set when the frame is delivered to the application. A minimum value of 4 should be set.

43.5.34 Transmit FIFO Section Empty Threshold (ENET_TSEM)

Address: 0h base + 1A0h offset = 1A0h



ENET_TSEM field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ENET_TSEM field descriptions (continued)

Field	Description
24–31 TX_SECTION_ EMPTY	Value Of The Transmit FIFO Section Empty Threshold Value, in 64-bit words, of the transmit FIFO section empty threshold. See Transmit FIFO for more information.

43.5.35 Transmit FIFO Almost Empty Threshold (ENET_TAEM)

Address: 0h base + 1A4h offset = 1A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								TX_ALMOST_EMPTY							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

ENET_TAEM field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TX_ALMOST_ EMPTY	Value of Transmit FIFO Almost Empty Threshold Value, in 64-bit words, of the transmit FIFO almost empty threshold. When the FIFO level reaches the value programmed in this field, and no end-of-frame is available for the frame, the MAC transmit logic, to avoid FIFO underflow, stops reading the FIFO and transmits a frame with an MII error indication. See Transmit FIFO for more information. A minimum value of 4 should be set.

43.5.36 Transmit FIFO Almost Full Threshold (ENET_TAFL)

Address: 0h base + 1A8h offset = 1A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																								TX_ALMOST_FULL							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

ENET_TAFL field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 TX_ALMOST_ FULL	Value Of The Transmit FIFO Almost Full Threshold

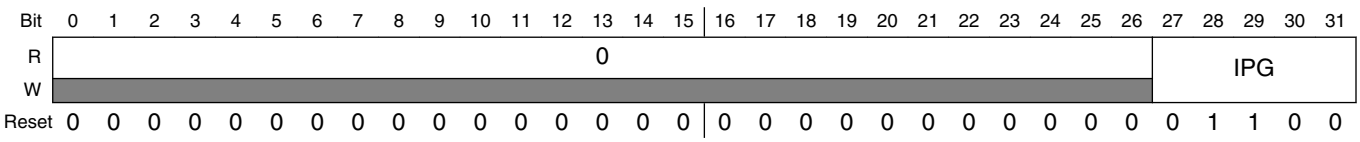
Table continues on the next page...

ENET_TAFL field descriptions (continued)

Field	Description
	Value, in 64-bit words, of the transmit FIFO almost full threshold. A minimum value of six is required . A recommended value of at least 8 should be set allowing a latency of two clock cycles to the application. If more latency is required the value can be increased as necessary (latency = TAFL - 5). When the FIFO level comes close to the maximum, so that there is no more space for at least TX_ALMOST_FULL number of words, the pin ff_tx_rdy is deasserted. If the application does not react on this signal, the FIFO write control logic, to avoid FIFO overflow, truncates the current frame and sets the error status. As a result, the frame will be transmitted with an GMII/MII error indication. See Transmit FIFO for more information. **NOTE:** A FIFO overflow is a fatal error and requires a global reset on the transmit datapath or at least deassertion of ETHEREN.

43.5.37 Transmit Inter-Packet Gap (ENET_TIPG)

Address: 0h base + 1ACh offset = 1ACh

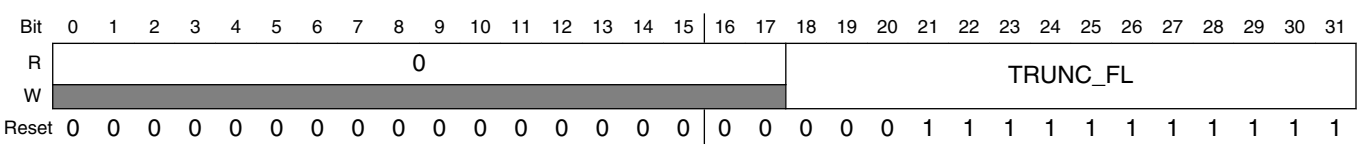


ENET_TIPG field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 IPG	Transmit Inter-Packet Gap Indicates the IPG, in bytes, between transmitted frames. Valid values range from 8 to 26. If the written value is less than 8 or greater than 26, the internal (effective) IPG is 12. **NOTE:** The IPG value read will be the value that was written, even if it is out of range.

43.5.38 Frame Truncation Length (ENET_FTRL)

Address: 0h base + 1B0h offset = 1B0h



ENET_FTRL field descriptions

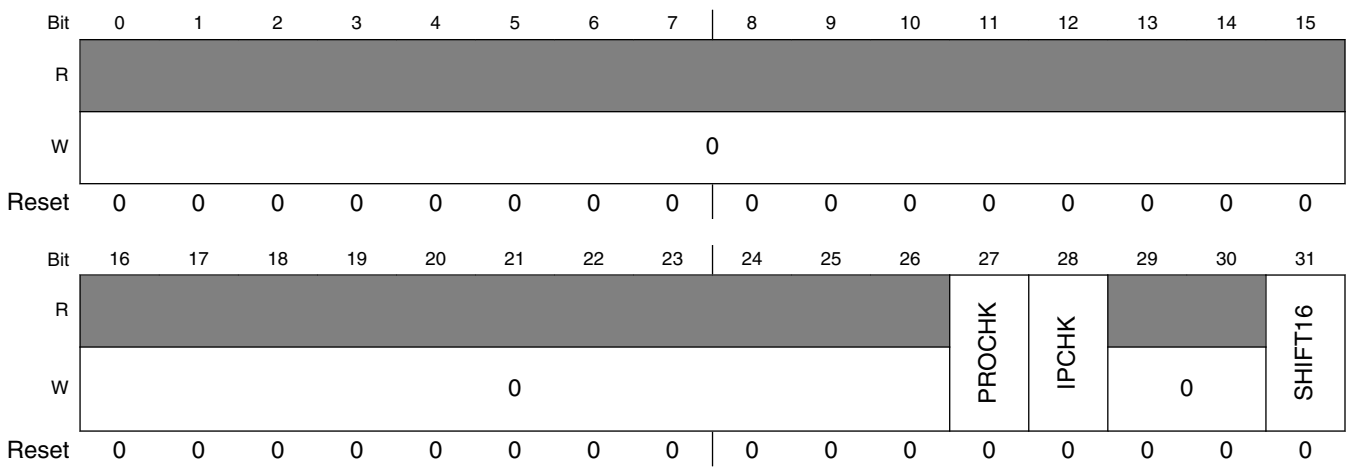
Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–31 TRUNC_FL	Frame Truncation Length Indicates the value a receive frame is truncated, if it is greater than this value. Must be greater than or equal to RCR[MAX_FL]. **NOTE:** Truncation happens at TRUNC_FL. However, when truncation occurs, the application (FIFO) may receive less data, guaranteeing that it never receives more than the set limit.

43.5.39 Transmit Accelerator Function Configuration (ENET_TACC)

TACC controls accelerator actions when sending frames. The register can be changed before or after each frame, but it must remain unmodified during frame writes into the transmit FIFO.

The TFWR[STRFWD] field must be set to use the checksum feature.

Address: 0h base + 1C0h offset = 1C0h



ENET_TACC field descriptions

Field	Description
0–26 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
27 PROCHK	Enables insertion of protocol checksum. 0 Checksum not inserted. 1 If an IP frame with a known protocol is transmitted, the checksum is inserted automatically into the frame. The checksum field must be cleared. The other frames are not modified.
28 IPCHK	Enables insertion of IP header checksum.

Table continues on the next page...

ENET_TACC field descriptions (continued)

Field	Description
	0 Checksum is not inserted. 1 If an IP frame is transmitted, the checksum is inserted automatically. The IP header checksum field must be cleared. If a non-IP frame is transmitted the frame is not modified.
29–30 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
31 SHIFT16	TX FIFO Shift-16 0 Disabled. 1 Indicates to the transmit data FIFO that the written frames contain two additional octets before the frame data. This means the actual frame begins at bit 16 of the first word written into the FIFO. This function allows putting the frame payload on a 32-bit boundary in memory, as the 14-byte Ethernet header is extended to a 16-byte header.

43.5.40 Receive Accelerator Function Configuration (ENET_RACC)

Address: 0h base + 1C4h offset = 1C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R									SHIFT16	LINEDIS				PRODIS	IPDIS	PADREM
W	0										0					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RACC field descriptions

Field	Description
0–23 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
24 SHIFT16	RX FIFO Shift-16 When this field is set, the actual frame data starts at bit 16 of the first word read from the RX FIFO aligning the Ethernet payload on a 32-bit boundary. **NOTE:** This function only affects the FIFO storage and has no influence on the statistics, which use the actual length of the frame received. 0 Disabled. 1 Instructs the MAC to write two additional bytes in front of each frame received into the RX FIFO.

Table continues on the next page...

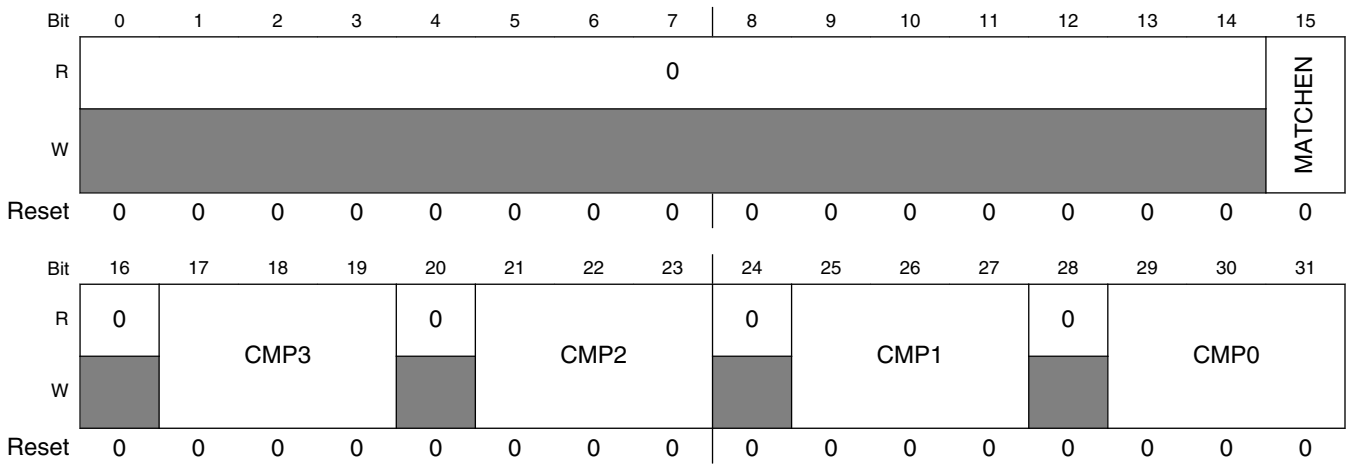
ENET_RACC field descriptions (continued)

Field	Description
25 LINEDIS	Enable Discard Of Frames With MAC Layer Errors 0 Frames with errors are not discarded. 1 Any frame received with a CRC, length, or PHY error is automatically discarded and not forwarded to the user application interface.
26–28 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
29 PRODIS	Enable Discard Of Frames With Wrong Protocol Checksum 0 Frames with wrong checksum are not discarded. 1 If a TCP/IP, UDP/IP, or ICMP/IP frame is received that has a wrong TCP, UDP, or ICMP checksum, the frame is discarded. Discarding is only available when the RX FIFO operates in store and forward mode (RSFL cleared).
30 IPDIS	Enable Discard Of Frames With Wrong IPv4 Header Checksum 0 Frames with wrong IPv4 header checksum are not discarded. 1 If an IPv4 frame is received with a mismatching header checksum, the frame is discarded. IPv6 has no header checksum and is not affected by this setting. Discarding is only available when the RX FIFO operates in store and forward mode (RSFL cleared).
31 PADREM	Enable Padding Removal For Short IP Frames 0 Padding not removed. 1 Any bytes following the IP payload section of the frame are removed from the frame.

43.5.41 Receive Classification Match Register for Class n (ENET_RCMR*n*)

This match register allows specifying up to four priorities, which are tested (OR'ed) simultaneously. The match detection uses the extracted VLAN field according to the rules for VLAN detection configured through the ECR register. If both match registers, RCMR1 and RCMR2, report a match at the same time, only the class 1 match is indicated as the final result.

Address: 0h base + 1C8h offset + (4d × i), where i=0d to 1d



ENET_RCMR*n* field descriptions

Field	Description
0–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 MATCHEN	Match Enable **NOTE:** A comparison is done only on incoming VLAN frames. If no VLAN frame is received no match will occur. If both match registers have overlapping compare values and hence can match both on the same frame, only class 1 will be indicated and the class 2 match is ignored. 0 Disabled (default): no compares will occur and the classification indicator for this class will never assert. 1 The register contents are valid and a comparison with all compare values is done when a VLAN frame is received.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–19 CMP3	Compare 3 Fourth value to compare against. If unused it must be set to the same value as CMP0.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ENET_RCMR*n* field descriptions (continued)

Field	Description
21–23 CMP2	Compare 2 Third value to compare against. If unused it must be set to the same value as CMP0.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–27 CMP1	Compare 1 Second value to compare against. If unused it must be set to the same value as CMP0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 CMP0	Compare 0 A three-bit value that will be compared with the frame's VLAN priority field (if a VLAN frame is received). All four compare values, CMP0..3, will be used in parallel. If any of the values match, a match for the class is reported (if MATCHEN is 1). **NOTE:** To implement a single priority match, all four compare values must be set to the same value.

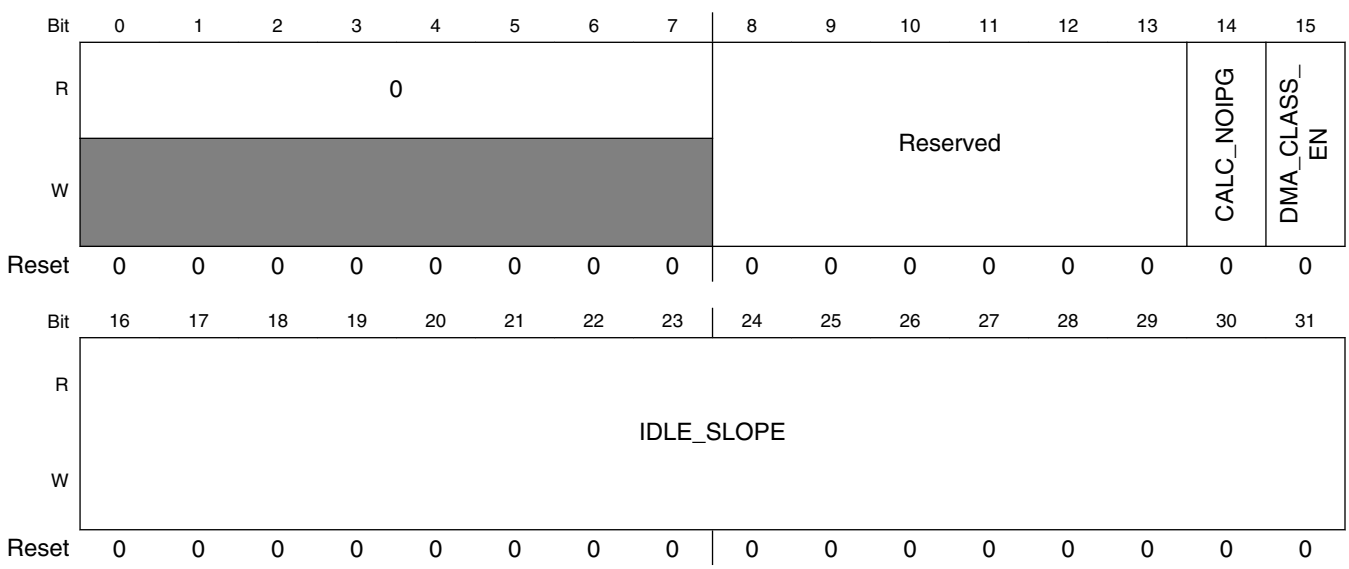
43.5.42 DMA Class Based Configuration (ENET_DMA*n*CFG)

The DMA class based configuration registers are used to configure the DMA controller interface to support the additional class 1 (buffer descriptor ring 1) and class 2 (buffer descriptor ring 2) traffic and define configuration options such as bandwidth allocation as needed.

NOTE

The registers are cleared when ECR[ETHEREN] becomes 0.

Address: 0h base + 1D8h offset + (4d × i), where i=0d to 1d



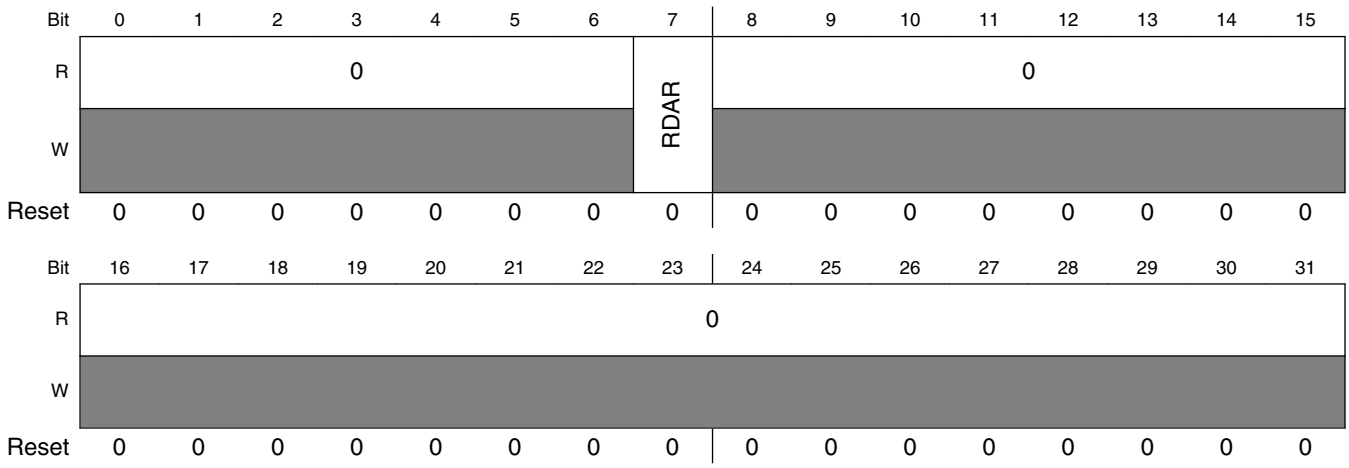
ENET_DMA*n*CFG field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–13 Reserved	This field is reserved. **NOTE:** Write only zeroes to this field.
14 CALC_NOIPG	Calculate no IPG Disable inclusion of IPG bytes for bandwidth calculations. 0 The traffic shaper function should consider 12 octets of IPG in addition to the frame data transferred for a frame when doing bandwidth calculations. This is the default. 1 Addition of 12 bytes for the IPG should be omitted when calculating the bandwidth (for traffic shaping, when writing a frame into the transmit FIFO, the shaper will usually consider 12 bytes of IPG for every frame as part of the bandwidth allocated by the frame. This addition can be suppressed, meaning short frames will become more bandwidth than large frames due to the relation of data to IPG overhead).
15 DMA_CLASS_EN	DMA class enable 0 The DMA controller's channel for the class is not used. **NOTE:** Disabling the DMA controller of a class also requires disabling the class match comparator for the class (see registers RCMRn). When class 1 and class 2 queues are disabled then their frames will be placed in queue 0. 1 Enable the DMA controller to support the corresponding descriptor ring for this class of traffic.
16–31 IDLE_SLOPE	Idle slope 16-bit value to define the per class idle slope setting used by the credit based shaper defining allocated bandwidth for the class. This value is used to calculate the BW (bandwidth) fraction, given by the equation, BW fraction = 1/(1+512/IDLE_SLOPE). Idle slope is restricted to certain values. For values less than 128, idle slope = 2^n, where n = 0, 1, 2, ...6. For values equal to or greater than 128, idle slope = 128×m, where m = 1, 2, 3, ...12. Example 1. BW fraction = 0.20 = 1/(1+(512/128)); therefore idleslope = 128. Example 2. BW fraction = 0.33 = 1/(1+(512/256)); therefore idleslope = 256. Example 3. BW fraction = 0.75 = 1/(1+(512/1536)); therefore idleslope = 1536. **NOTE:** For AVB applications, the BW fraction of class 1 and class 2 combined must not exceed .75.

43.5.43 Receive Descriptor Active Register - Ring 1 (ENET_RDAR1)

RDAR1 is a command register, written by the user, to indicate that the receive descriptor ring 1 has been updated, that is, that the driver produced empty receive buffers with the empty bit set.

Address: 0h base + 1E0h offset = 1E0h



ENET_RDAR1 field descriptions

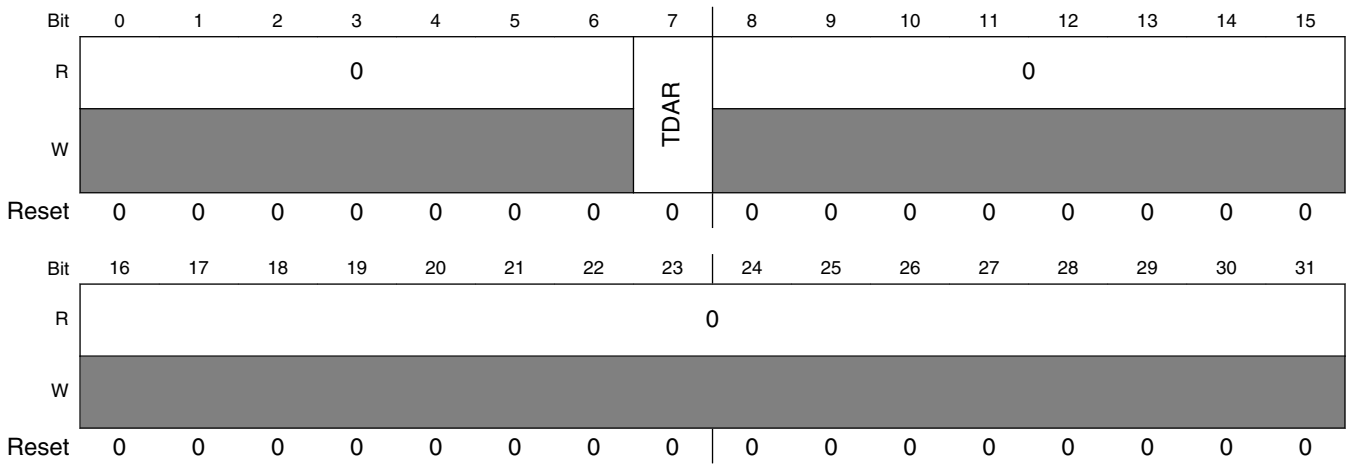
Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 RDAR	Receive Descriptor Active Always set to 1 when this register is written, regardless of the value written. This field is cleared by the MAC device when no additional empty descriptors remain in the receive ring. It is also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.44 Transmit Descriptor Active Register - Ring 1 (ENET_TDAR1)

TDAR1 is a command register that the user writes to indicate that transmit descriptor ring 1 has been updated, that is, that transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor.

The TDAR register is cleared at reset, when ECR[ETHEREN] transitions from set to cleared, or when ECR[RESET] is set.

Address: 0h base + 1E4h offset = 1E4h



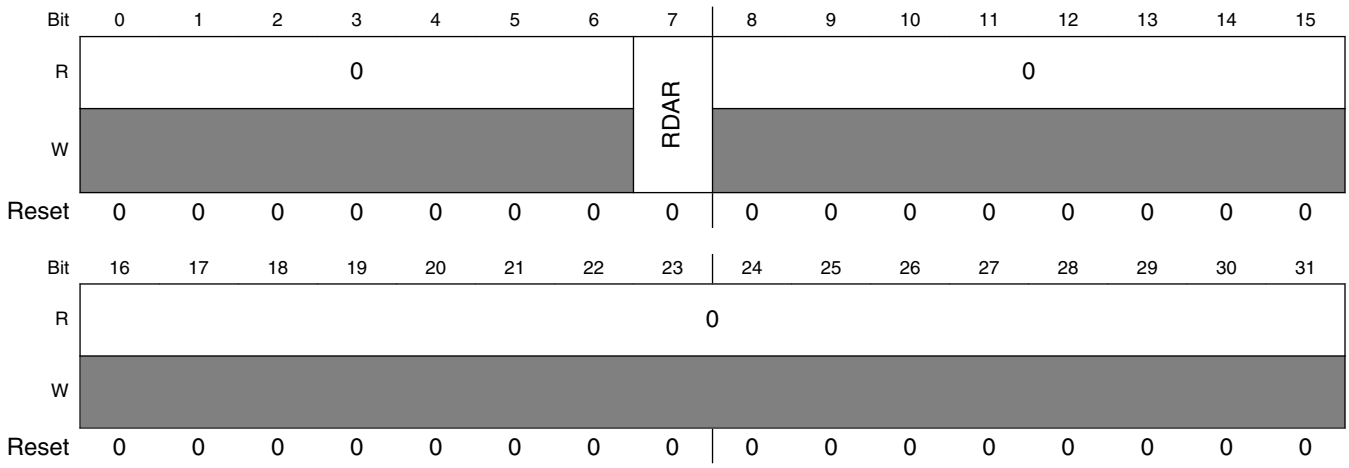
ENET_TDAR1 field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TDAR	Transmit Descriptor Active Always set to 1 when this register is written, regardless of the value written. This bit is cleared by the MAC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.45 Receive Descriptor Active Register - Ring 2 (ENET_RDAR2)

RDAR2 is a command register, written by the user, to indicate that the receive descriptor ring 2 has been updated, that is, that the driver produced empty receive buffers with the empty bit set.

Address: 0h base + 1E8h offset = 1E8h



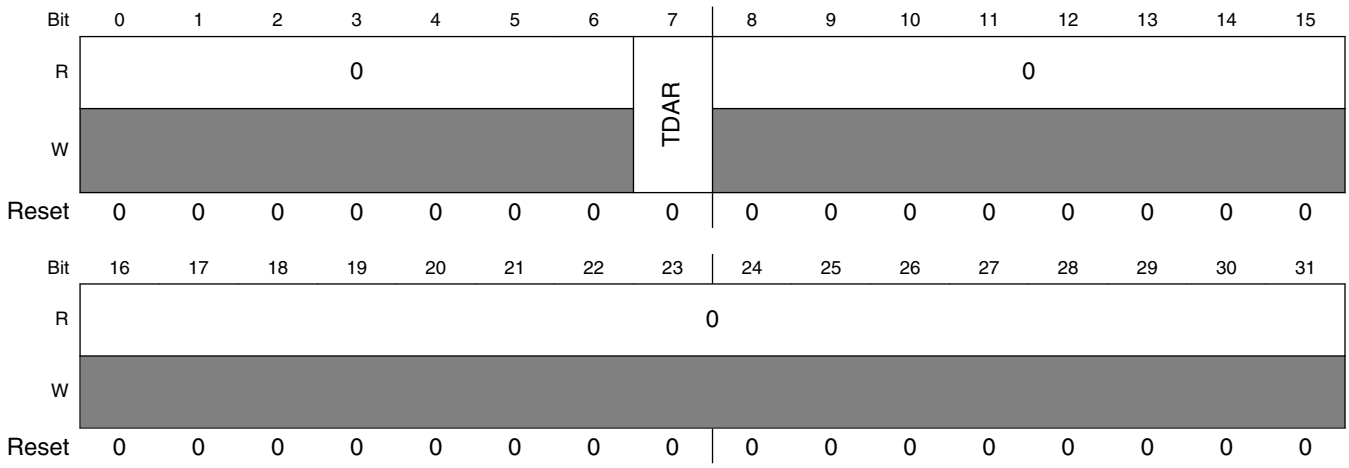
ENET_RDAR2 field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 RDAR	Receive Descriptor Active Always set to 1 when this register is written, regardless of the value written. This field is cleared by the MAC device when no additional empty descriptors remain in the receive ring. It is also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.46 Transmit Descriptor Active Register - Ring 2 (ENET_TDAR2)

TDAR2 is a command register that the user writes to indicate that transmit descriptor ring 2 has been updated, that is, that transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor.

Address: 0h base + 1ECh offset = 1ECh



ENET_TDAR2 field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
7 TDAR	Transmit Descriptor Active Always set to 1 when this register is written, regardless of the value written. This bit is cleared by the MAC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

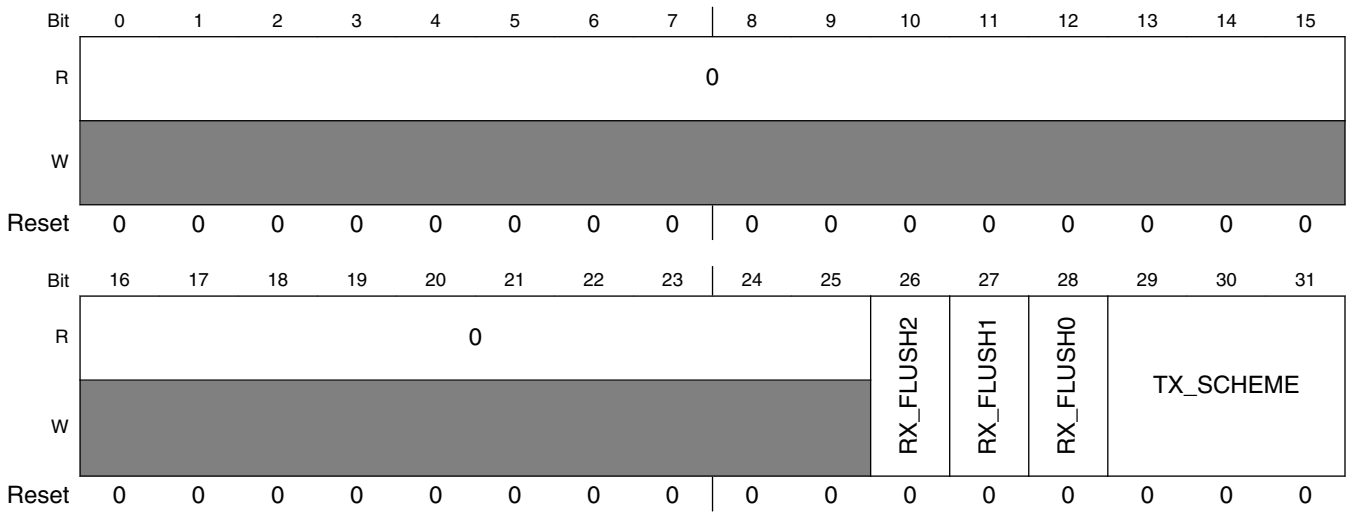
43.5.47 QOS Scheme (ENET_QOS)

This register sets the QOS scheme.

NOTE

When both class 1 and class 2 are disabled, RX flushing for these rings must also be disabled.

Address: 0h base + 1F0h offset = 1F0h

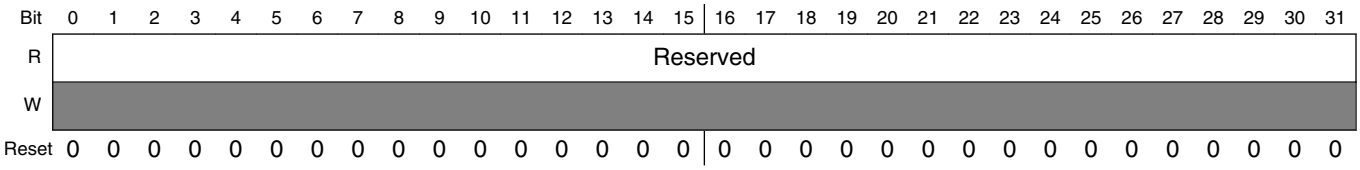


ENET_QOS field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 RX_FLUSH2	RX Flush Ring 2 Enable or disable RX Flush for ring 2. See Receive flushing. 0 Disable 1 Enable
27 RX_FLUSH1	RX Flush Ring 1 Enable or disable RX Flush for ring 1. See Receive flushing. 0 Disable 1 Enable
28 RX_FLUSH0	RX Flush Ring 0 Enable or disable RX Flush for ring 0. See Receive flushing. 0 Disable 1 Enable
29–31 TX_SCHEME	TX scheme configuration Configuration information for DMA to select transmitter queue selection/arbitration scheme. 000 Credit-based scheme 001 Round-robin scheme 010-111 Reserved

43.5.48 Reserved Statistic Register (ENET_RMON_T_DROP)

Address: 0h base + 200h offset = 200h

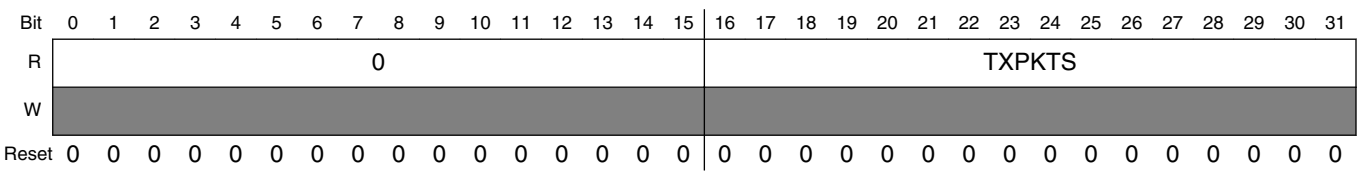


ENET_RMON_T_DROP field descriptions

Field	Description
0–31 Reserved	This read-only field always has the value 0. This field is reserved.

43.5.49 Tx Packet Count Statistic Register (ENET_RMON_T_PACKETS)

Address: 0h base + 204h offset = 204h



ENET_RMON_T_PACKETS field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Packet count Transmit packet count

43.5.50 Tx Broadcast Packets Statistic Register (ENET_RMON_T_BC_PKT)

RMON Tx Broadcast Packets

Address: 0h base + 208h offset = 208h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_BC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Broadcast packets

43.5.51 Tx Multicast Packets Statistic Register (ENET_RMON_T_MC_PKT)

Address: 0h base + 20Ch offset = 20Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_MC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Multicast packets

43.5.52 Tx Packets with CRC/Align Error Statistic Register (ENET_RMON_T_CRC_ALIGN)

Address: 0h base + 210h offset = 210h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_CRC_ALIGN field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Packets with CRC/align error

43.5.53 Tx Packets Less Than Bytes and Good CRC Statistic Register (ENET_RMON_T_UNDERSIZE)

Address: 0h base + 214h offset = 214h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_UNDERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets less than 64 bytes with good CRC

43.5.54 Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (ENET_RMON_T_OVERSIZE)

Address: 0h base + 218h offset = 218h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_OVERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets greater than MAX_FL bytes with good CRC

43.5.55 Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_T_FRAG)

.

Address: 0h base + 21Ch offset = 21Ch

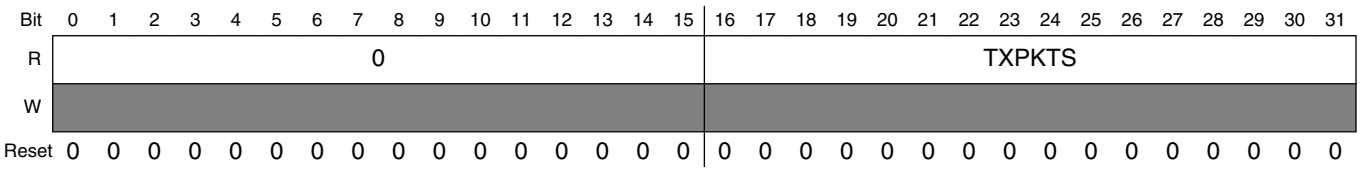
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_FRAG field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of packets less than 64 bytes with bad CRC

43.5.56 Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (ENET_RMON_T_JAB)

Address: 0h base + 220h offset = 220h

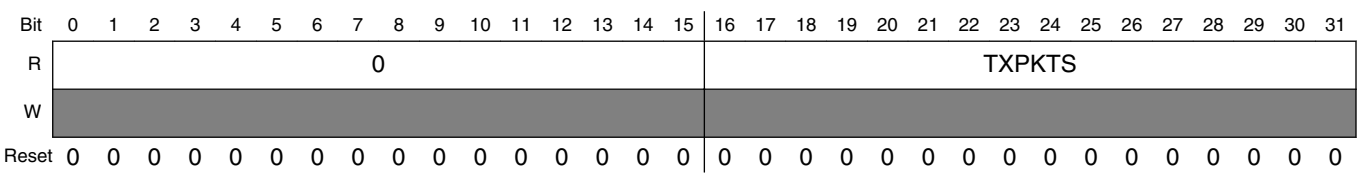


ENET_RMON_T_JAB field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets greater than MAX_FL bytes and bad CRC

43.5.57 Tx Collision Count Statistic Register (ENET_RMON_T_COL)

Address: 0h base + 224h offset = 224h



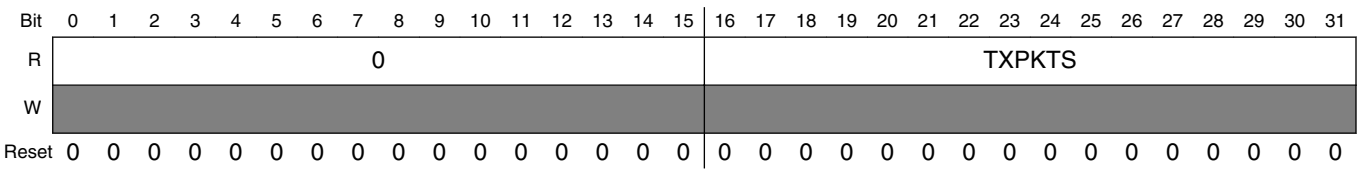
ENET_RMON_T_COL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit collisions

43.5.58 Tx 64-Byte Packets Statistic Register (ENET_RMON_T_P64)

.

Address: 0h base + 228h offset = 228h

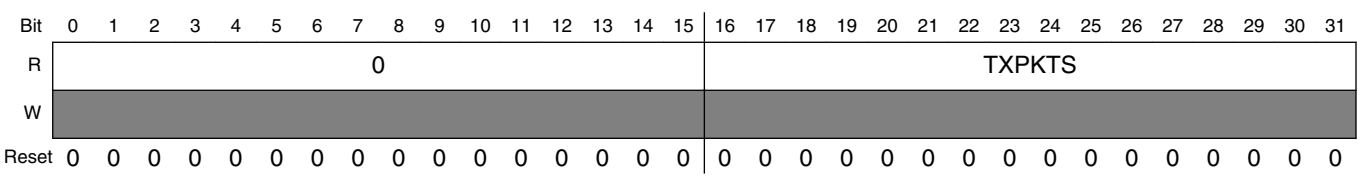


ENET_RMON_T_P64 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 64-byte transmit packets

43.5.59 Tx 65- to 127-byte Packets Statistic Register (ENET_RMON_T_P65TO127)

Address: 0h base + 22Ch offset = 22Ch



ENET_RMON_T_P65TO127 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 65- to 127-byte transmit packets

43.5.60 Tx 128- to 255-byte Packets Statistic Register (ENET_RMON_T_P128TO255)

Address: 0h base + 230h offset = 230h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_T_P128TO255 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 128- to 255-byte transmit packets

43.5.61 Tx 256- to 511-byte Packets Statistic Register (ENET_RMON_T_P256TO511)

Address: 0h base + 234h offset = 234h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																TXPKTS															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

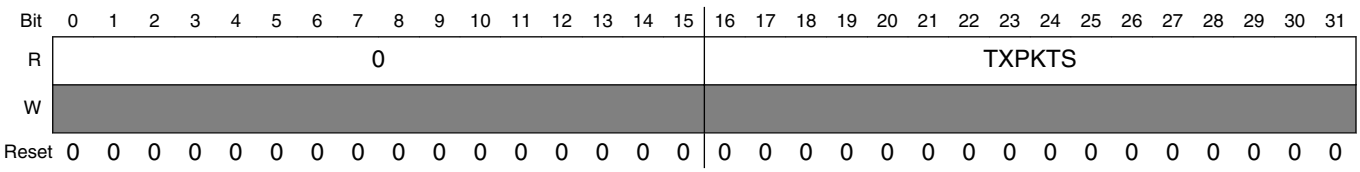
ENET_RMON_T_P256TO511 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 256- to 511-byte transmit packets

43.5.62 Tx 512- to 1023-byte Packets Statistic Register (ENET_RMON_T_P512TO1023)

.

Address: 0h base + 238h offset = 238h

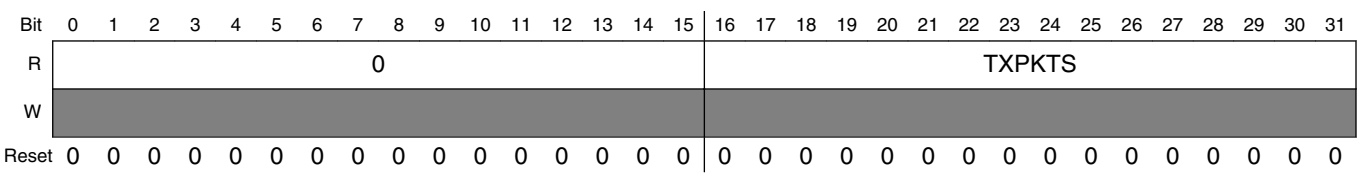


ENET_RMON_T_P512TO1023 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 512- to 1023-byte transmit packets

43.5.63 Tx 1024- to 2047-byte Packets Statistic Register (ENET_RMON_T_P1024TO2047)

Address: 0h base + 23Ch offset = 23Ch

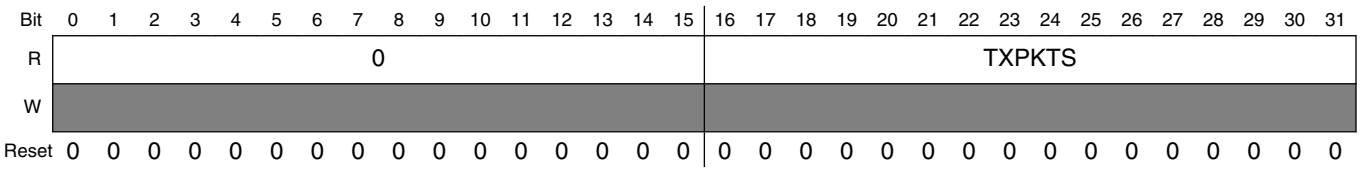


ENET_RMON_T_P1024TO2047 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of 1024- to 2047-byte transmit packets

43.5.64 Tx Packets Greater Than 2048 Bytes Statistic Register (ENET_RMON_T_P_GTE2048)

Address: 0h base + 240h offset = 240h

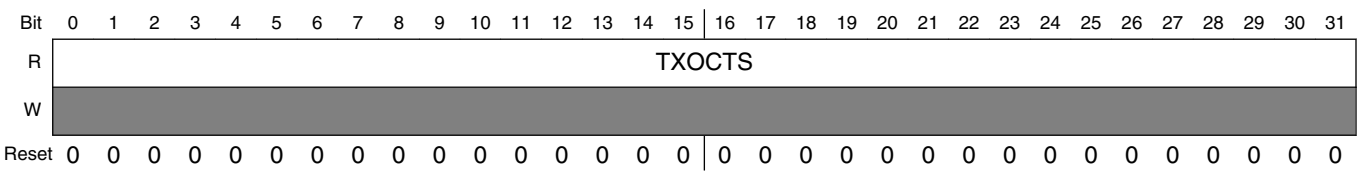


ENET_RMON_T_P_GTE2048 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TXPKTS	Number of transmit packets greater than 2048 bytes

43.5.65 Tx Octets Statistic Register (ENET_RMON_T_OCTETS)

Address: 0h base + 244h offset = 244h

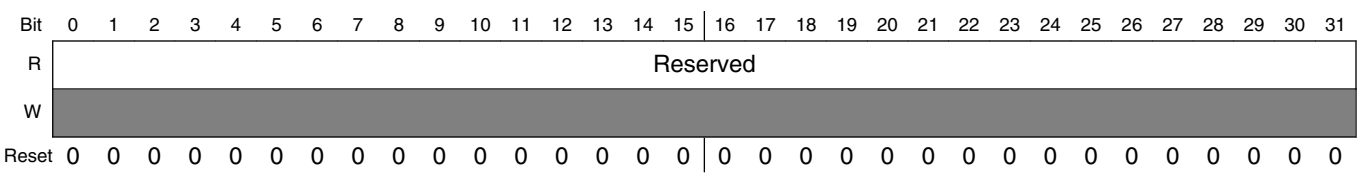


ENET_RMON_T_OCTETS field descriptions

Field	Description
0–31 TXOCTS	Number of transmit octets

43.5.66 Reserved Statistic Register (ENET_IEEE_T_DROP)

Address: 0h base + 248h offset = 248h

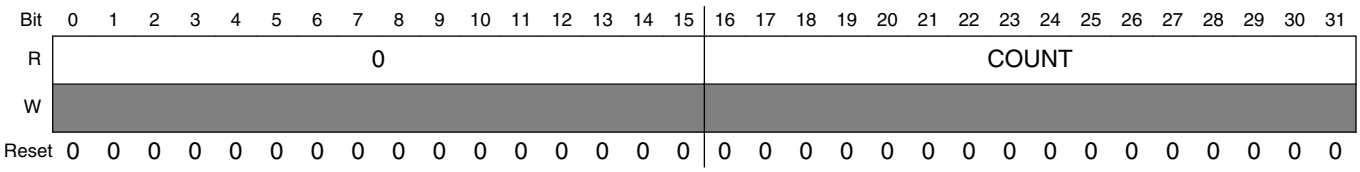


ENET_IEEE_T_DROP field descriptions

Field	Description
0–31 Reserved	This read-only field always has the value 0. This field is reserved.

43.5.67 Frames Transmitted OK Statistic Register (ENET_IEEE_T_FRAME_OK)

Address: 0h base + 24Ch offset = 24Ch

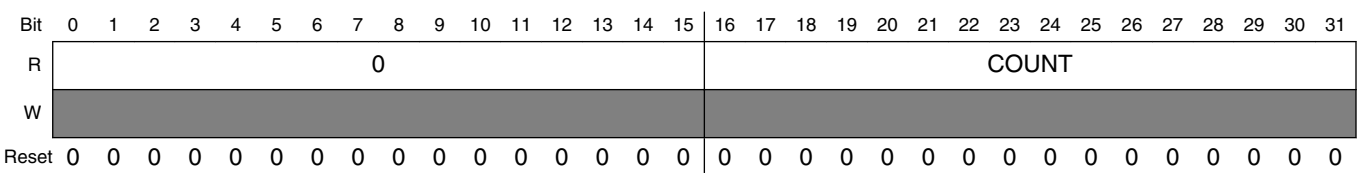


ENET_IEEE_T_FRAME_OK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted OK **NOTE:** Does not increment for the broadcast frames when broadcast reject is enabled and promiscuous mode is disabled within the receive control register (RCR).

43.5.68 Frames Transmitted with Single Collision Statistic Register (ENET_IEEE_T_1COL)

Address: 0h base + 250h offset = 250h



ENET_IEEE_T_1COL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with one collision

43.5.69 Frames Transmitted with Multiple Collisions Statistic Register (ENET_IEEE_T_MCOL)

Address: 0h base + 254h offset = 254h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_MCOL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with multiple collisions

43.5.70 Frames Transmitted after Deferral Delay Statistic Register (ENET_IEEE_T_DEF)

Address: 0h base + 258h offset = 258h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_DEF field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with deferral delay

43.5.71 Frames Transmitted with Late Collision Statistic Register (ENET_IEEE_T_LCOL)

Address: 0h base + 25Ch offset = 25Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_LCOL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with late collision

43.5.72 Frames Transmitted with Excessive Collisions Statistic Register (ENET_IEEE_T_EXCOL)

Address: 0h base + 260h offset = 260h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_EXCOL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with excessive collisions

43.5.73 Frames Transmitted with Tx FIFO Underrun Statistic Register (ENET_IEEE_T_MACERR)

Address: 0h base + 264h offset = 264h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_MACERR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with transmit FIFO underrun

43.5.74 Frames Transmitted with Carrier Sense Error Statistic Register (ENET_IEEE_T_CSERR)

Address: 0h base + 268h offset = 268h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_CSERR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames transmitted with carrier sense error

43.5.75 Reserved Statistic Register (ENET_IEEE_T_SQE)

Address: 0h base + 26Ch offset = 26Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_SQE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	This read-only field is reserved and always has the value 0. **NOTE:** Counter not implemented as no SQE information is available.

43.5.76 Flow Control Pause Frames Transmitted Statistic Register (ENET_IEEE_T_FDXFC)

Address: 0h base + 270h offset = 270h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_FDXFC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of flow-control pause frames transmitted

43.5.77 Octet Count for Frames Transmitted w/o Error Statistic Register (ENET_IEEE_T_OCTETS_OK)

Address: 0h base + 274h offset = 274h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_T_OCTETS_OK field descriptions

Field	Description
0–31 COUNT	Octet count for frames transmitted without error **NOTE** Counts total octets (includes header and FCS fields).

43.5.78 Rx Packet Count Statistic Register (ENET_RMON_R_PACKETS)

Address: 0h base + 284h offset = 284h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_PACKETS field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of packets received

43.5.79 Rx Broadcast Packets Statistic Register (ENET_RMON_R_BC_PKT)

Address: 0h base + 288h offset = 288h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_BC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive broadcast packets

43.5.80 Rx Multicast Packets Statistic Register (ENET_RMON_R_MC_PKT)

Address: 0h base + 28Ch offset = 28Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_MC_PKT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive multicast packets

43.5.81 Rx Packets with CRC/Align Error Statistic Register (ENET_RMON_R_CRC_ALIGN)

Address: 0h base + 290h offset = 290h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_CRC_ALIGN field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets with CRC or align error

43.5.82 Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (ENET_RMON_R_UNDERSIZE)

Address: 0h base + 294h offset = 294h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_UNDERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets with less than 64 bytes and good CRC

43.5.83 Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (ENET_RMON_R_OVERSIZE)

Address: 0h base + 298h offset = 298h

Bit	0–15	16–31
R	0	COUNT
W		
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ENET_RMON_R_OVERSIZE field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets greater than MAX_FL and good CRC

43.5.84 Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (ENET_RMON_R_FRAG)

Address: 0h base + 29Ch offset = 29Ch

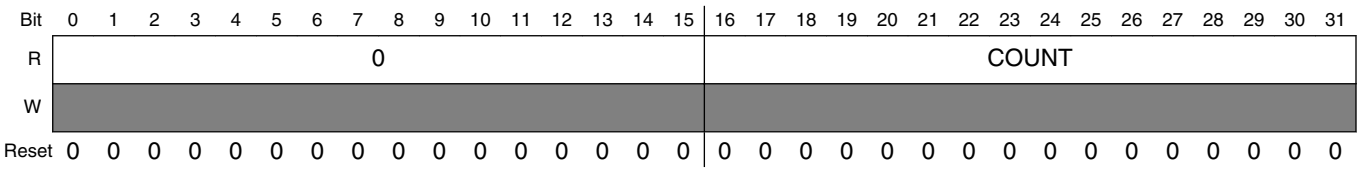
Bit	0–15	16–31
R	0	COUNT
W		
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ENET_RMON_R_FRAG field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets with less than 64 bytes and bad CRC

43.5.85 Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (ENET_RMON_R_JAB)

Address: 0h base + 2A0h offset = 2A0h

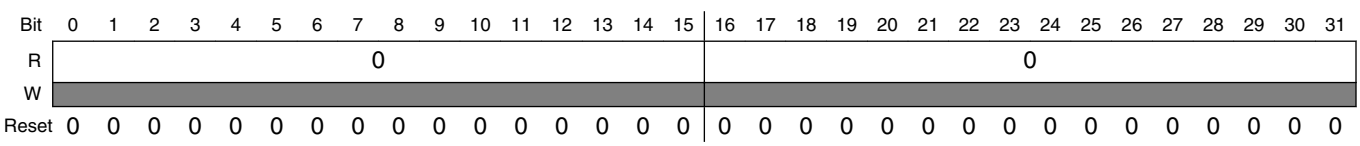


ENET_RMON_R_JAB field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of receive packets greater than MAX_FL and bad CRC

43.5.86 Reserved Statistic Register (ENET_RMON_R_RESVD_0)

Address: 0h base + 2A4h offset = 2A4h

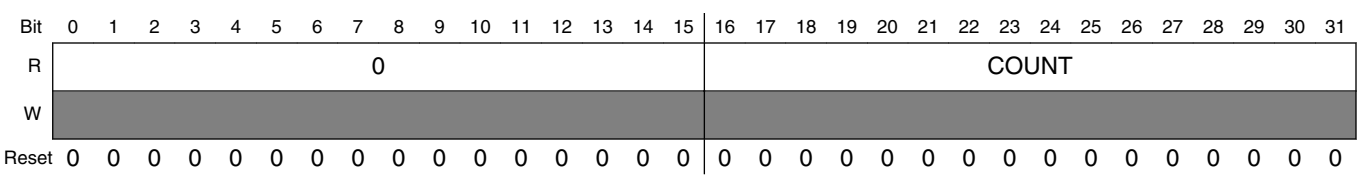


ENET_RMON_R_RESVD_0 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.87 Rx 64-Byte Packets Statistic Register (ENET_RMON_R_P64)

Address: 0h base + 2A8h offset = 2A8h



ENET_RMON_R_P64 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 64-byte receive packets

43.5.88 Rx 65- to 127-Byte Packets Statistic Register (ENET_RMON_R_P65TO127)

Address: 0h base + 2ACh offset = 2ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P65TO127 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 65- to 127-byte recieve packets

43.5.89 Rx 128- to 255-Byte Packets Statistic Register (ENET_RMON_R_P128TO255)

Address: 0h base + 2B0h offset = 2B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P128TO255 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 128- to 255-byte recieve packets

43.5.90 Rx 256- to 511-Byte Packets Statistic Register (ENET_RMON_R_P256TO511)

Address: 0h base + 2B4h offset = 2B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P256TO511 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 256- to 511-byte recieve packets

43.5.91 Rx 512- to 1023-Byte Packets Statistic Register (ENET_RMON_R_P512TO1023)

Address: 0h base + 2B8h offset = 2B8h

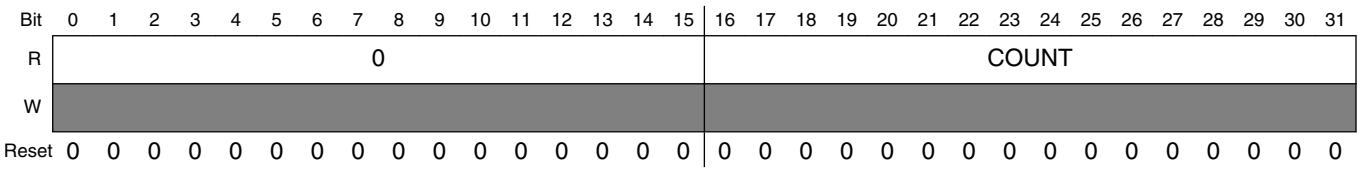
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RMON_R_P512TO1023 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 512- to 1023-byte recieve packets

43.5.92 Rx 1024- to 2047-Byte Packets Statistic Register (ENET_RMON_R_P1024TO2047)

Address: 0h base + 2BCh offset = 2BCh

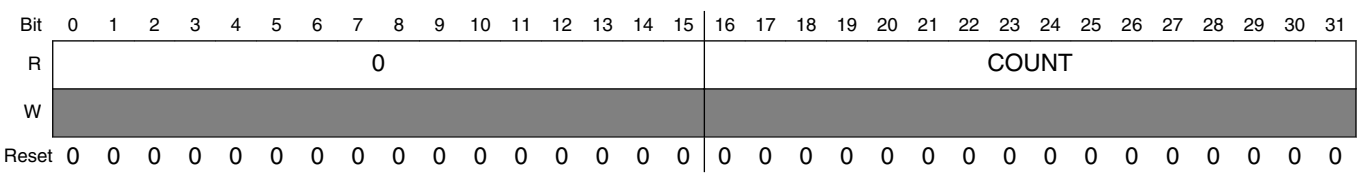


ENET_RMON_R_P1024TO2047 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of 1024- to 2047-byte recieve packets

43.5.93 Rx Packets Greater than 2048 Bytes Statistic Register (ENET_RMON_R_P_GTE2048)

Address: 0h base + 2C0h offset = 2C0h

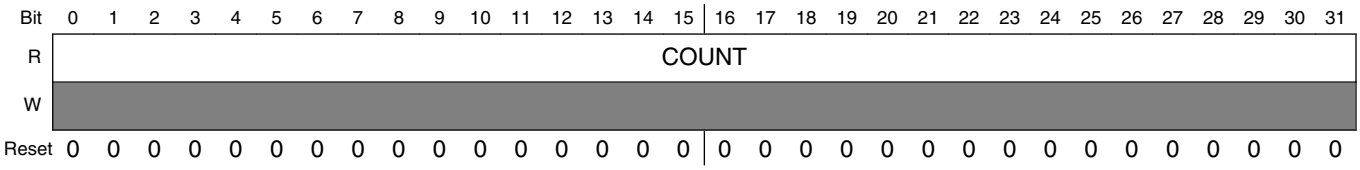


ENET_RMON_R_P_GTE2048 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of greater-than-2048-byte recieve packets

43.5.94 Rx Octets Statistic Register (ENET_RMON_R_OCTETS)

Address: 0h base + 2C4h offset = 2C4h



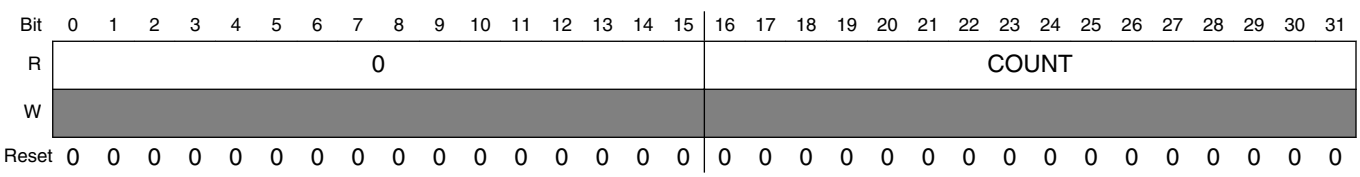
ENET_RMON_R_OCTETS field descriptions

Field	Description
0–31 COUNT	Number of receive octets

43.5.95 Frames not Counted Correctly Statistic Register (ENET_IEEE_R_DROP)

Counter increments if a frame with invalid or missing SFD character is detected and has been dropped. None of the other counters increments if this counter increments.

Address: 0h base + 2C8h offset = 2C8h



ENET_IEEE_R_DROP field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Frame count

43.5.96 Frames Received OK Statistic Register (ENET_IEEE_R_FRAME_OK)

Address: 0h base + 2CCh offset = 2CCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_FRAME_OK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames received OK

43.5.97 Frames Received with CRC Error Statistic Register (ENET_IEEE_R_CRC)

Address: 0h base + 2D0h offset = 2D0h

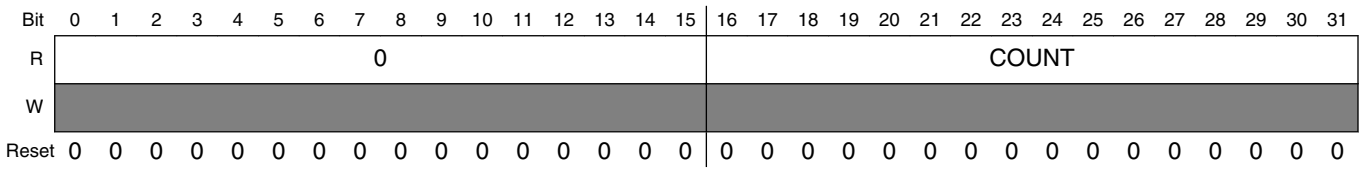
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_CRC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames received with CRC error

43.5.98 Frames Received with Alignment Error Statistic Register (ENET_IEEE_R_ALIGN)

Address: 0h base + 2D4h offset = 2D4h

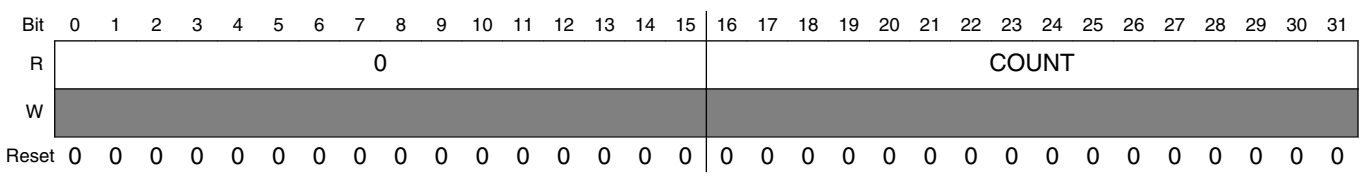


ENET_IEEE_R_ALIGN field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of frames received with alignment error

43.5.99 Receive FIFO Overflow Count Statistic Register (ENET_IEEE_R_MACERR)

Address: 0h base + 2D8h offset = 2D8h



ENET_IEEE_R_MACERR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Receive FIFO overflow count

43.5.100 Flow Control Pause Frames Received Statistic Register (ENET_IEEE_R_FDXFC)

Address: 0h base + 2DCh offset = 2DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_FDXFC field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Number of flow-control pause frames received

43.5.101 Octet Count for Frames Received without Error Statistic Register (ENET_IEEE_R_OCTETS_OK)

Address: 0h base + 2E0h offset = 2E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_IEEE_R_OCTETS_OK field descriptions

Field	Description
0–31 COUNT	Number of octets for frames received without error **NOTE:** Counts total octets (includes header and FCS fields). Does not increment for the broadcast frames when broadcast reject is enabled and promiscuous mode is disabled within the receive control register (RCR).

43.5.102 Adjustable Timer Control Register (ENET_ATCR)

ATCR command fields can trigger the corresponding events directly. It is not necessary to preserve any of the configuration fields when a command field is set in the register, that is, no read-modify-write is required.

NOTE

The CAPTURE and RESTART fields and bits 19 and 21 must be 0 in order to write to the other fields in this register.

Address: 0h base + 400h offset = 400h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		SLAVE	Reserved	CAPTURE	Reserved	RESTART		PINPER			PEREN	OFFRST	OFFEN		EN
W								0		0	1				0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATCR field descriptions

Field	Description
0–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18 SLAVE	Enable Timer Slave Mode 0 The timer is active and all configuration fields in this register are relevant. 1 The internal timer is disabled and the externally provided timer value is used. All other fields, except CAPTURE, in this register have no effect. CAPTURE can still be used to capture the current timer value.
19 Reserved	This field is reserved. Always write 0 to this field.
20 CAPTURE	Capture Timer Value When this field is set, all other fields are ignored during a write. This field automatically clears to 0 after the command completes. 0 No effect. 1 The current time is captured and can be read from the ATVR register.
21 Reserved	This field is reserved. Always write 0 to this field.
22 RESTART	Reset Timer Resets the timer to zero. This has no effect on the counter enable. If the counter is enabled when this field is set, the timer is reset to zero and starts counting from there. When set, all other fields are ignored during a write. This field automatically clears to 0 after the command completes.
23 Reserved	This field is reserved.
24 PINPER	Enables event signal output assertion on period event. **NOTE:** Not all devices contain the event signal output. See the chip configuration details.

Table continues on the next page...

ENET_ATCR field descriptions (continued)

Field	Description
	0 Disable. 1 Enable.
25 Reserved	This field is reserved.
26 Reserved	This field is reserved. **NOTE:** This field must be written always with one.
27 PEREN	Enable Periodical Event 0 Disable. 1 A period event interrupt can be generated (EIR[TS_TIMER]) and the event signal output is asserted when the timer wraps around according to the periodic setting ATPER. The timer period value must be set before setting this bit. **NOTE:** Not all devices contain the event signal output. See the chip configuration details.
28 OFFRST	Reset Timer On Offset Event 0 The timer is not affected and no action occurs, besides clearing OFFEN, when the offset is reached. 1 If OFFEN is set, the timer resets to zero when the offset setting is reached. The offset event does not cause a timer interrupt.
29 OFFEN	Enable One-Shot Offset Event 0 Disable. 1 The timer can be reset to zero when the given offset time is reached (offset event). The field is cleared when the offset event is reached, so no further event occurs until the field is set again. The timer offset value must be set before setting this field.
30 Reserved	This field is reserved.
31 EN	Enable Timer 0 The timer stops at the current value. 1 The timer starts incrementing.

43.5.103 Timer Value Register (ENET_ATVR)

Address: 0h base + 404h offset = 404h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ATIME																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATVR field descriptions

Field	Description
0–31 ATIME	A write sets the timer. A read returns the last captured value. To read the current value, issue a capture command (i.e., set ATCR[CAPTURE]) prior to reading this register.

43.5.104 Timer Offset Register (ENET_ATOFF)

Address: 0h base + 408h offset = 408h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	OFFSET																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATOFF field descriptions

Field	Description
0–31 OFFSET	Offset value for one-shot event generation. When the timer reaches the value, an event can be generated to reset the counter. If the increment value in ATINC is given in true nanoseconds, this value is also given in true nanoseconds.

43.5.105 Timer Period Register (ENET_ATPER)

Address: 0h base + 40Ch offset = 40Ch

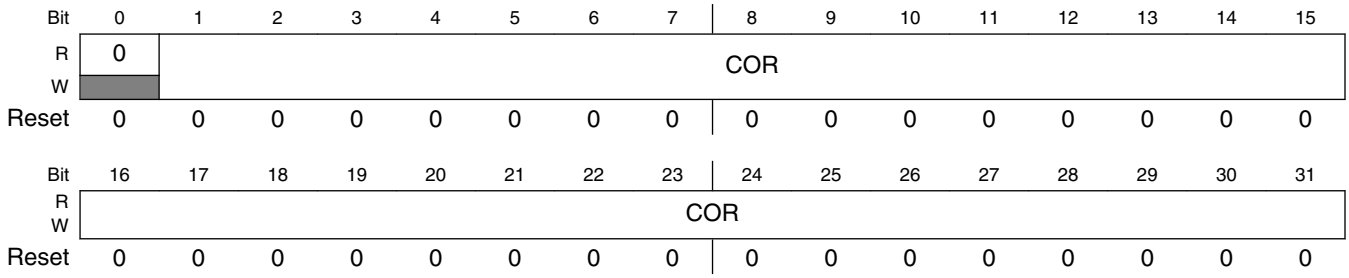
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PERIOD																															
Reset	0	0	1	1	1	0	1	1	1	0	0	1	1	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0

ENET_ATPER field descriptions

Field	Description
0–31 PERIOD	Value for generating periodic events. Each instance the timer reaches this value, the period event occurs and the timer restarts. If the increment value in ATINC is given in true nanoseconds, this value is also given in true nanoseconds. The value should be initialized to 1,000,000,000 (1×10^9) to represent a timer wrap around of one second. The increment value set in ATINC should be set to the true nanoseconds of the period of clock ts_clk, hence implementing a true 1 second counter. **NOTE:** The value of PERIOD has the following constraint: 2^{32} – ENET_ATINC[INC_COR] – 3×ENET_ATINC[INC] ≥ PERIOD > 0.

43.5.106 Timer Correction Register (ENET_ATCOR)

Address: 0h base + 410h offset = 410h

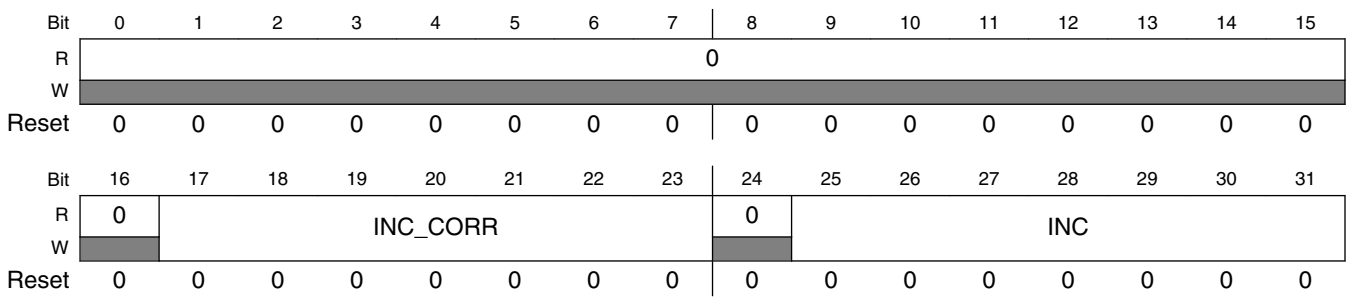


ENET_ATCOR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–31 COR	Correction Counter Wrap-Around Value Defines after how many timer clock cycles (ts_clk) the correction counter should be reset and trigger a correction increment on the timer. The amount of correction is defined in ATINC[INC_CORR]. A value of 0 disables the correction counter and no corrections occur. **NOTE:** This value is given in clock cycles, not in nanoseconds as all other values.

43.5.107 Time-Stamping Clock Period Register (ENET_ATINC)

Address: 0h base + 414h offset = 414h



ENET_ATINC field descriptions

Field	Description
0–16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–23 INC_CORR	Correction Increment Value This value is added every time the correction timer expires (every clock cycle given in ATCOR). A value less than INC slows down the timer. A value greater than INC speeds up the timer.

Table continues on the next page...

ENET_ATINC field descriptions (continued)

Field	Description
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 INC	Clock Period Of The Timestamping Clock (ts_clk) In Nanoseconds The timer increments by this amount each clock cycle. For example, set to 10 for 100 MHz, 8 for 125 MHz, 5 for 200 MHz. **NOTE:** For highest precision, use a value that is an integer fraction of the period set in ATPER.

43.5.108 Timestamp of Last Transmitted Frame (ENET_ATSTMP)

Address: 0h base + 418h offset = 418h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TIMESTAMP																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_ATSTMP field descriptions

Field	Description
0–31 TIMESTAMP	Timestamp of the last frame transmitted by the core that had TxBD[TS] set . This register is only valid when EIR[TS_AVAIL] is set.

43.5.109 Pattern Match Data Register (ENET_MDATA)

See Frame parser for more information.

Address: 0h base + 580h offset = 580h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MATCHDATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_MDATA field descriptions

Field	Description
0–31 MATCHDATA	Match Data Data to be used for pattern matching when corresponding MMASK bit is set to 1.

43.5.110 Match Entry Mask Register (ENET_MMASK)

See Frame parser for more information.

Address: 0h base + 584h offset = 584h

Bit	0–15	16–31
R W	MATCHMASK	
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ENET_MMASK field descriptions

Field	Description
0–31 MATCHMASK	Match Mask Mask used in conjunction with MDATA register to control which bits in MDATA to compare. If MMASK[*n*] is set to 1, then MDATA[*n*] will be used for comparison. If MMASK[*n*] is 0, then MDATA[*n*] will not be compared.

43.5.111 Match Entry Rules Configuration Register (ENET_MCONFIG)

See Frame parser for more information.

Address: 0h base + 588h offset = 588h

Bit	0	1	2	3–7	8–15
R W	AF	RF	IM	0	OK_INDEX
Reset	0	0	0	0 0 0 0 0	0 0 0 0 0 0 0 0

Bit	16–23	24–29	30	31
R W	0	FRMOFF	0	
Reset	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0	0

ENET_MCONFIG field descriptions

Field	Description
0 AF	Accept Frame If okay, put frame into queue defined by the OK_INDEX. Meaning of okay is defined by the IM field. **NOTE:** If both the AF and RF fields are set to 1, AF will take precedence and RF will be ignored.
1 RF	Reject Frame

Table continues on the next page...

ENET_MCONFIG field descriptions (continued)

Field	Description
	If okay, reject/drop frame. The meaning of okay is defined by the IM field. **NOTE:** If both the AF and RF fields are set to 1, AF will take precedence and RF will be ignored.
2 IM	Invert Match When set to 1, okay means not matched. When cleared to 0, okay means match with entry.
3–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 OK_INDEX	When AF = 0 and RF = 0, this value shows the next entry of the matching table to be used for comparison instead of using the next entry sequentially. This value should be less than the total number of valid entries defined by RXPCTL[MAX_INDEX]. The minimum value can be 0. When AF = 1, this value is used as the queue index where the frame should be received. The least-significant two bits of this field are used as the queue index. Maximum value of the queue index is 2.
16–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–29 FRMOFF	Frame Offset Points to the word, from the start of the frame, to be compared for matching. This value must be less than or equal to MAXFRMOFF or else a parser programming error will occur, resulting in ENET_EIR[PARSERR] being asserted and the frame being discarded.
30–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

43.5.112 Match Entry Read/Write Command Register (ENET_MENTRYRW)

See Frame parser for more information.

Address: 0h base + 58Ch offset = 58Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						RD	WR	ENTRYADD							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_MENTRYRW field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

ENET_MENTRYRW field descriptions (continued)

Field	Description
22 RD	Entry Read Command When set to 1, the current entry values of the Match Entry table are read and available within the match entry registers, MATCH_DATA, MATCH_MASK, and MCONFIG. This field is automatically cleared when the read operation is completed. Match entry registers are valid and ready for reading only when this field is cleared to 0 after a read operation. **NOTE:** If both the WR and RD fields are set to 1, the write operation will take precedence and the read operation will be ignored.
23 WR	Entry write command When set to 1, the current values of the match entry registers (MATCH_DATA, MATCH_MASK, and MCONFIG) are written to the Match Entry table. This field is automatically cleared when the write operation is complete. **NOTE:** If both the WR and RD fields are set to 1, the write operation will take precedence and the read operation will be ignored.
24–31 ENTRYADD	Entry Address Address location of the Match Entry table on which read or write operations will be done. The Match Entry table contains ENET_RXPCTL[MAXINDEX] entries of 96-bit words. Therefore valid ENTRYADD values range from a minimum of 0 to a maximum of ENET_RXPCTL[MAXINDEX]-1. For example, if ENET_RXPCTL[MAXINDEX] = 64, then ENTRYADD may range from 0 to 63, inclusive.

43.5.113 Receive Parser Control Register (ENET_RXPCTL)

See Frame parser for more information.

Address: 0h base + 590h offset = 590h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R								ACPTEERR	ENDERRQ							
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MAXINDEX											PRSRSCLR			INVBYTORD	ENPARSER
W									0				0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_RXPCTL field descriptions

Field	Description
0–6 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
7 ACPTEERR	Accept End Error Accept the current frame when parsing is not completed for the current frame and parsing for the next frame has started. The corresponding accept statistics counter for the queue defined by ENDERRQ will be incremented. **NOTE:** RXPARST[PRSENDERR] assertion is independent of this field.
8–15 ENDERRQ	End Error Queue Queue number for the accepted frames that encounters the RXPARST[PRSENDERR]. Only one bit should be set. 00000001 Place the frame in Queue 0 00000010 Place the frame in Queue 1 00000100 Place the frame in Queue 2
16–23 MAXINDEX	Maximum Index Number of valid entries in the Match Array table. This value must be less than or equal to the depth of the physical table and greater than or equal to 1. If this value is cleared to 0, parser is disabled. **NOTE:** The depth of the Match Array table is chip-specific. See the ENET chip-specific information for this value.
24–26 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
27 PRSRSCLR	Clear Parser Statistics Counter The field is automatically cleared to 0 after resetting the parser statistics register. Returns 0 on read.
28–29 Reserved	This field is reserved. This write-only field is reserved. It must always be written with the value 0.
30 INVBYTORD	Inverse Frame Byte Order When set to 1: • The byte order within a frame word is changed before comparison. • The least-significant byte is compared with most-significant byte of the parser table entry data word. • And most-significant byte of the frame word is compared with least-significant byte of the table entry data word.
31 ENPARSER	Enable Receive Parser **NOTE:** When enabled, at least one entry within the Match Array Table should be valid. See MAXINDEX field for more details. 0 Parser is disabled. 1 Parser is enabled.

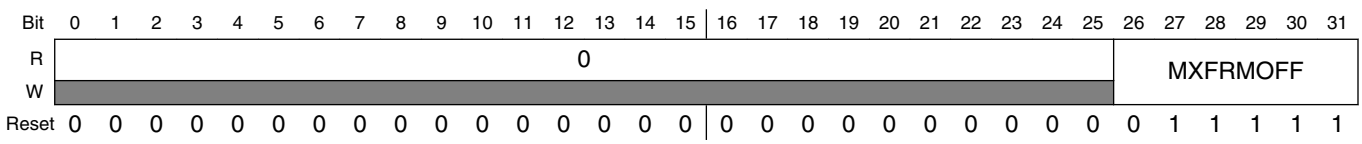
43.5.114 Maximum Frame Offset (ENET_MAXFRMOFF)

Maximum value of the 32-bit word index position of the frame to be inspected. Maximum value is 63. With a value of 31 (default) it is possible to inspect any byte position from 0 to 127.

NOTE

MCONFIG[FRMOFF] should be always less than or equal to this value, otherwise parser state machine will generate parser programming error, assert EIR[PARS_ERR], and frame will be discarded.

Address: 0h base + 594h offset = 594h

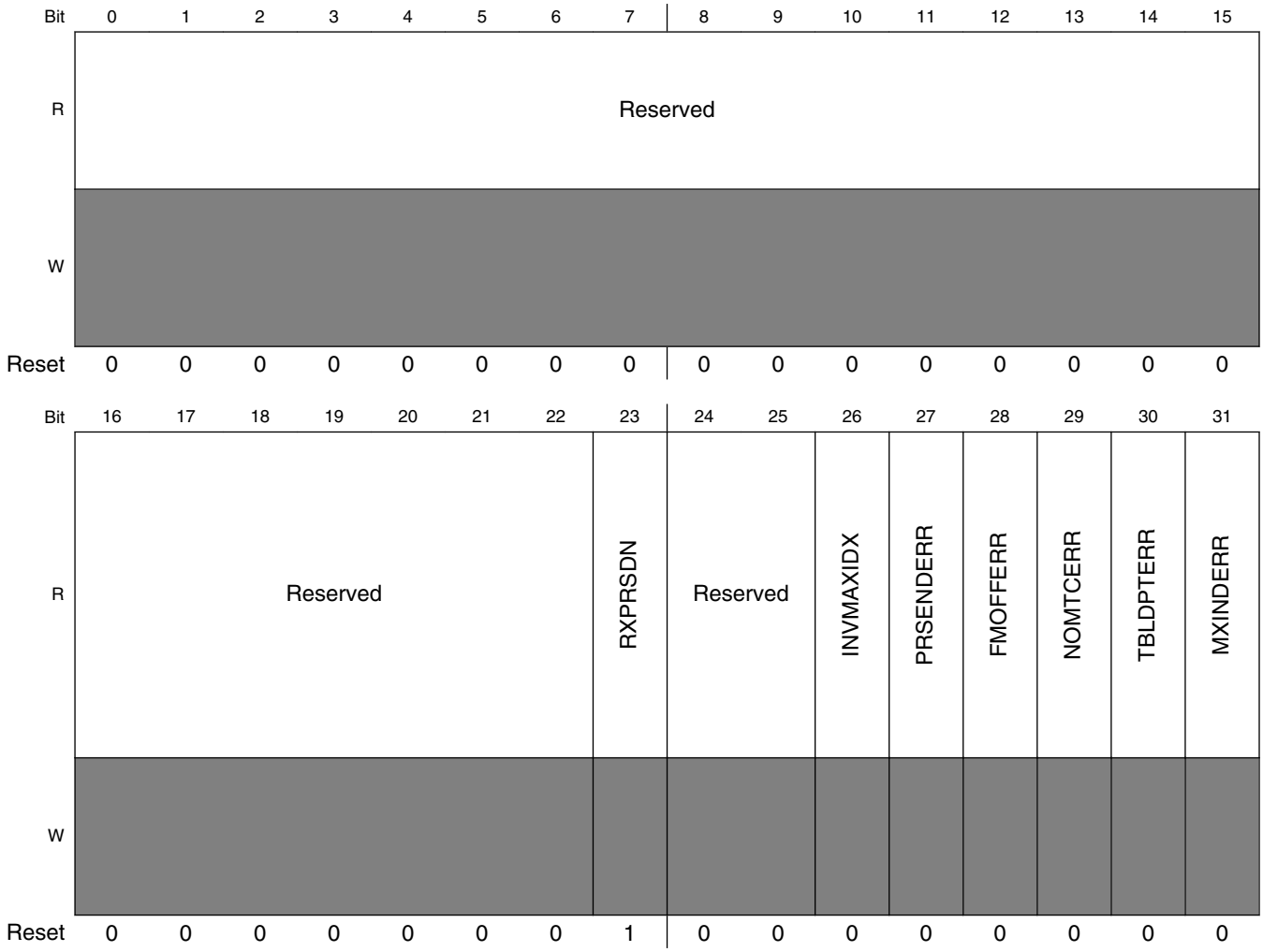


ENET_MAXFRMOFF field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 MXFRMOFF	Max. Frame Offset

43.5.115 Receive Parser Status (ENET_RXPARST)

Address: 0h base + 598h offset = 598h



ENET_RXPARST field descriptions

Field	Description
0–22 Reserved	This field is reserved.
23 RXPRSDN	Receive Parser Done Current status of the receive parser state machine. Asserted to indicate that the parser is in idle state. De-asserted when parser is active and currently parsing a receive frame.
24–25 Reserved	This field is reserved.
26 INVMAXIDX	Invalid Value of MAXINDEX This field is set to 1 when the parser is currently enabled, but the number of valid entries (MAXINDEX) within the table is set to 0 or is larger than the physical table depth.

Table continues on the next page...

ENET_RXPARST field descriptions (continued)

Field	Description
27 PRSENDERR	Parser End Error Frame parsing started for the next frame before end of parsing for the current frame. The frame is accepted or rejected depending on the configuration of RXPCTL[ACPTEERR]. This field is cleared on read or if parser is disabled.
28 FMOFFERR	Maximum Frame Offset Error Frame rejected due to parser error as frame offset is larger than expected maximum frame offset value defined by the MAXFRMOFF register. Latch high bit. Cleared on read or if parser is disabled.
29 NOMTCERR	No Match Error No Match error. Frame rejected due to parser error as no match or invert match found after end of parsing. Latch high bit. Cleared on read or if parser is disabled.
30 TBLDPTERR	Table Depth Error Frame rejected due to parser error as table entry index larger than or equal to parser table physical depth. Latch high bit. Cleared on read or if parser is disabled.
31 MXINDERR	Maximum Index Error Frame rejected due to parser error as table entry index greater than or equal to number of valid entries within the table defined by RXPCTL[MAX_INDEX]. Latch high bit. Cleared on read or if parser is disabled.

43.5.116 Parser Discard Count (ENET_PARSDSCD)

Number of frames rejected by the receive parser due to no matching, inverse matching, or parser programming error.

Address: 0h base + 5A0h offset = 5A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_PARSDSCD field descriptions

Field	Description
0–31 COUNT	Count

43.5.117 Parser Accept Count 0 (ENET_PRSACPT0)

Number of frames accepted by the receive parser for queue 0.

Address: 0h base + 5A4h offset = 5A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_PRSACPT0 field descriptions

Field	Description
0–31 COUNT	Count

43.5.118 Parser Reject Count 0 (ENET_PRSRJCT0)

Number of frames rejected by the receive parser for queue 0.

NOTE

This count does not include the rejected frames due to a no-matching, inverse-matching, or parser-programming error.

Address: 0h base + 5A8h offset = 5A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

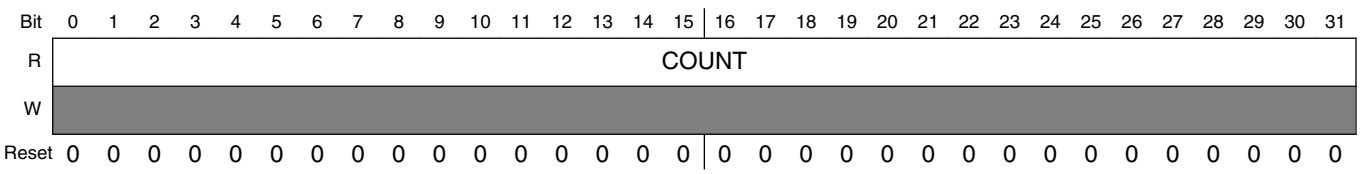
ENET_PRSRJCT0 field descriptions

Field	Description
0–31 COUNT	Count

43.5.119 Parser Accept Count 1 (ENET_PRSACPT1)

Number of frames accepted by the receive parser for queue 1.

Address: 0h base + 5ACh offset = 5ACh



ENET_PRSACPT1 field descriptions

Field	Description
0–31 COUNT	Count

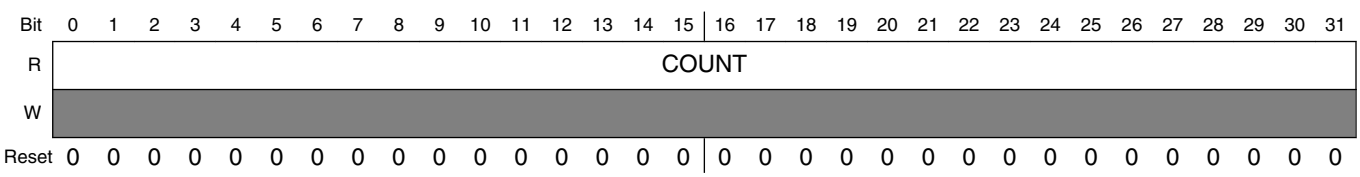
43.5.120 Parser Reject Count 1 (ENET_PRSRJCT1)

Number of frames rejected by the receive parser for queue 1.

NOTE

This count does not include the rejected frames due to a no-matching, inverse-matching, or parser-programming error.

Address: 0h base + 5B0h offset = 5B0h



ENET_PRSRJCT1 field descriptions

Field	Description
0–31 COUNT	Count

43.5.121 Parser Accept Count 2 (ENET_PRSACPT2)

Number of frames accepted by the receive parser for queue 2.

Address: 0h base + 5B4h offset = 5B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_PRSACPT2 field descriptions

Field	Description
0–31 COUNT	Count

43.5.122 Parser Reject Count 2 (ENET_PRSRJCT2)

Number of frames rejected by the receive parser for queue 2.

NOTE

This count does not include the rejected frames due to a no-matching, inverse-matching, or parser-programming error.

Address: 0h base + 5B8h offset = 5B8h

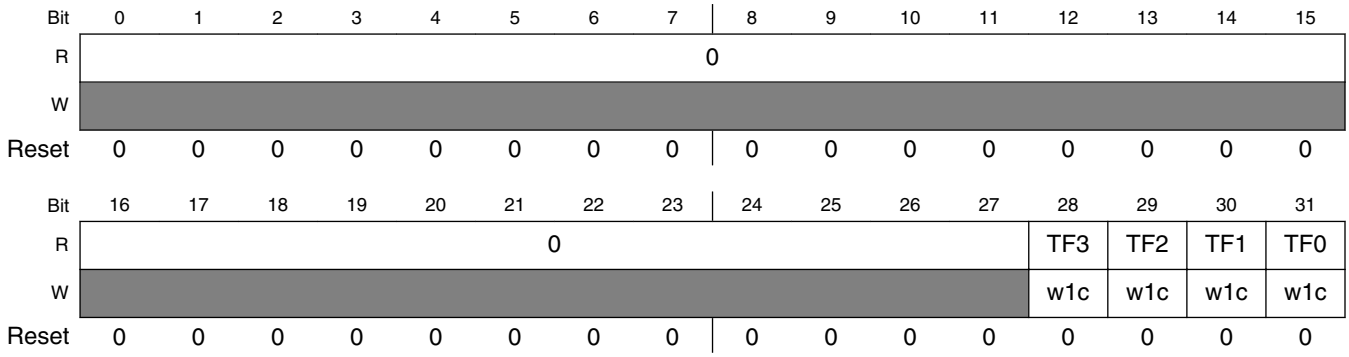
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_PRSRJCT2 field descriptions

Field	Description
0–31 COUNT	Count

43.5.123 Timer Global Status Register (ENET_TGSR)

Address: 0h base + 604h offset = 604h

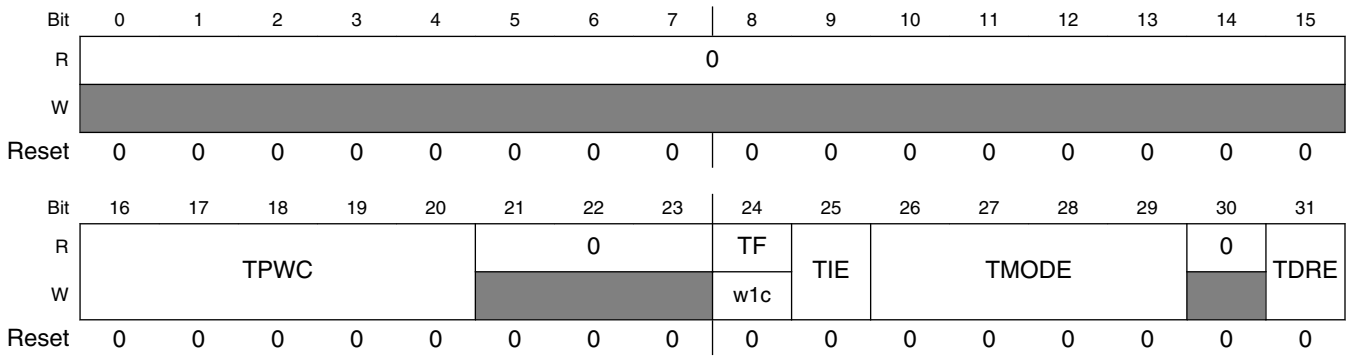


ENET_TGSR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 TF3	Copy Of Timer Flag For Channel 3 0 Timer Flag for Channel 3 is clear 1 Timer Flag for Channel 3 is set
29 TF2	Copy Of Timer Flag For Channel 2 0 Timer Flag for Channel 2 is clear 1 Timer Flag for Channel 2 is set
30 TF1	Copy Of Timer Flag For Channel 1 0 Timer Flag for Channel 1 is clear 1 Timer Flag for Channel 1 is set
31 TF0	Copy Of Timer Flag For Channel 0 0 Timer Flag for Channel 0 is clear 1 Timer Flag for Channel 0 is set

43.5.124 Timer Control Status Register (ENET_TCSR*n*)

Address: 0h base + 608h offset + (8d × i), where i=0d to 3d



ENET_TCSR*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–20 TPWC	Timer PulseWidth Control Specifies the pulse width associated with TMODE values of 1110 or 11X1. Updating this field takes a few cycles to register because it is synchronized to the 1588 clock. When changing this field: 1. Always disable the channel and read the TMODE field to verify that the channel is disabled. 2. Set TPWC to the desired value. 3. Reenable the channel. 00000 Pulse width is one 1588-clock cycle. 00001 Pulse width is two 1588-clock cycles. 00010 Pulse width is three 1588-clock cycles. 00011 Pulse width is four 1588-clock cycles. ... ... 11111 Pulse width is 32 1588-clock cycles.
21–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 TF	Timer Flag Sets when input capture or output compare occurs. This flag is double buffered between the module clock and 1588 clock domains. When this field is 1, it can be cleared to 0 by writing 1 to it. 0 Input Capture or Output Compare has not occurred. 1 Input Capture or Output Compare has occurred.
25 TIE	Timer Interrupt Enable 0 Interrupt is disabled 1 Interrupt is enabled
26–29 TMODE	Timer Mode

Table continues on the next page...

ENET_TCSR*n* field descriptions (continued)

Field	Description
	Updating the Timer Mode field takes a few cycles to register because it is synchronized to the 1588 clock. The version of Timer Mode returned on a read is from the 1588 clock domain. When changing Timer Mode, always disable the channel and read this register to verify the channel is disabled first. 0000 Timer Channel is disabled. 0001 Timer Channel is configured for Input Capture on rising edge. 0010 Timer Channel is configured for Input Capture on falling edge. 0011 Timer Channel is configured for Input Capture on both edges. 0100 Timer Channel is configured for Output Compare - software only. 0101 Timer Channel is configured for Output Compare - toggle output on compare. 0110 Timer Channel is configured for Output Compare - clear output on compare. 0111 Timer Channel is configured for Output Compare - set output on compare. 1000 Reserved 1010 Timer Channel is configured for Output Compare - clear output on compare, set output on overflow. 10X1 Timer Channel is configured for Output Compare - set output on compare, clear output on overflow. 110X Reserved 1110 Timer Channel is configured for Output Compare - pulse output low on compare for 1 to 32 1588-clock cycles as specified by TPWC. 1111 Timer Channel is configured for Output Compare - pulse output high on compare for 1 to 32 1588-clock cycles as specified by TPWC.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TDRE	Timer DMA Request Enable 0 DMA request is disabled 1 DMA request is enabled

43.5.125 Timer Compare Capture Register (ENET_TCCR*n*)

Address: 0h base + 60Ch offset + (8d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TCC																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ENET_TCCR*n* field descriptions

Field	Description
0–31 TCC	Timer Capture Compare This register is double buffered between the module clock and 1588 clock domains. When configured for compare, the 1588 clock domain updates with the value in the module clock domain whenever the Timer Channel is first enabled and on each subsequent compare. Write to this register with the first compare value before enabling the Timer Channel. When the Timer Channel is enabled, write the second compare value either immediately, or at least before the first compare occurs. After each compare, write the next compare value before the previous compare occurs and before clearing the Timer Flag.

ENET_TCCR*n* field descriptions (continued)

Field	Description
	The compare occurs one 1588 clock cycle after the IEEE 1588 Counter increments past the compare value in the 1588 clock domain. If the compare value is less than the value of the 1588 Counter when the Timer Channel is first enabled, then the compare does not occur until following the next overflow of the 1588 Counter. If the compare value is greater than the IEEE 1588 Counter when the 1588 Counter overflows, or the compare value is less than the value of the IEEE 1588 Counter after the overflow, then the compare occurs one 1588 clock cycle following the overflow. When configured for capture, the value of the IEEE 1588 Counter is captured into the 1588 clock domain and then updated into the module clock domain, provided the Timer Flag is clear. Always read the capture value before clearing the Timer Flag. **NOTE:** TCC must fall within the following range: ENET_ATINC[INC] ≤ TCC ≤ (ENET_ATPER[PERIOD] – ENET_ATINC[INC])

43.6 Functional description

This section provides a complete functional description of the MAC-NET core.

43.6.1 Ethernet MAC frame formats

The IEEE 802.3 standard defines the Ethernet frame format as follows:

- Minimum length of 64 bytes
- Maximum length of 1518 bytes excluding the preamble and the start frame delimiter (SFD) bytes

An Ethernet frame consists of the following fields:

- Seven bytes preamble
- Start frame delimiter (SFD)
- Two address fields
- Length or type field
- Data field
- Frame check sequence (CRC value)

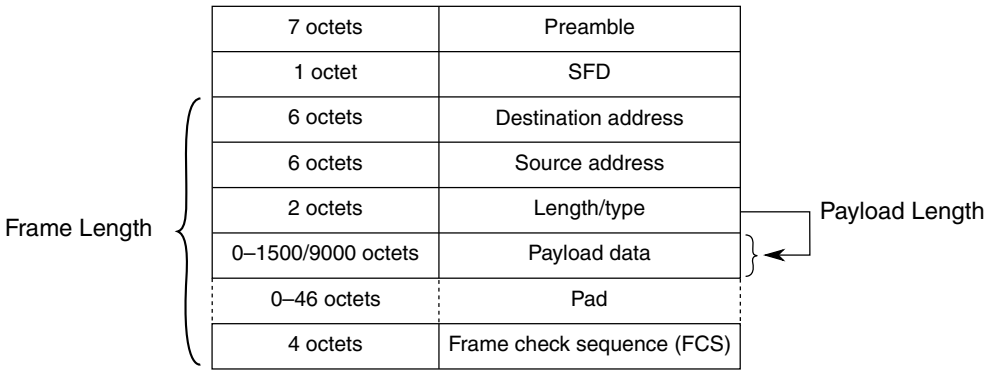


Figure 43-2. MAC frame format overview

Optionally, MAC frames can be VLAN-tagged with an additional four-byte field inserted between the MAC source address and the type/length field. VLAN tagging is defined by the IEEE P802.1q specification. VLAN-tagged frames have a maximum length of 1522 bytes, excluding the preamble and the SFD bytes.

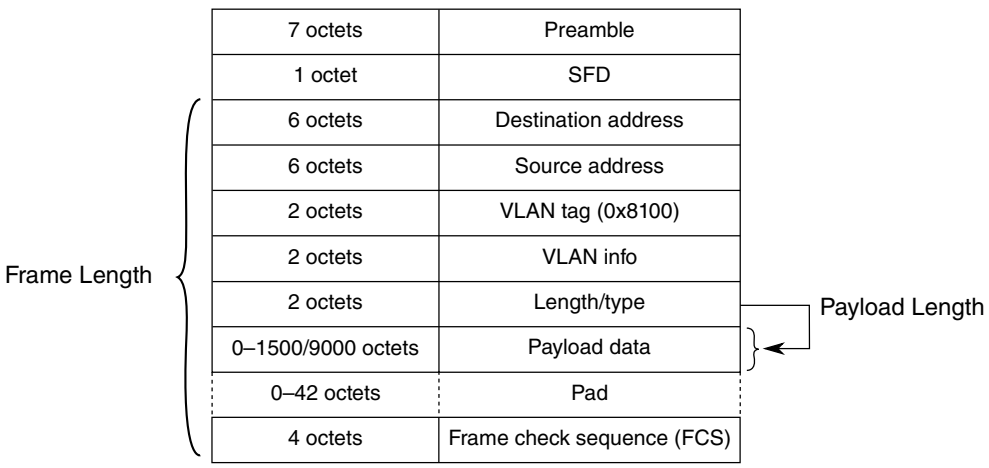


Figure 43-3. VLAN-tagged MAC frame format overview

Table 43-4. MAC frame definition

Term	Description
Frame length	Defines the length, in octets, of the complete frame without preamble and SFD. A frame has a valid length if it contains at least 64 octets and does not exceed the programmed maximum length.
Payload length	The length/type field indicates the length of the frame's payload section. The most significant byte is sent/received first. • If the length/type field is set to a value less than 46, the payload is padded so that the minimum frame length requirement (64 bytes) is met. For VLAN-tagged frames, a value less than 42 indicates a padded frame. • If the length/type field is set to a value larger than the programmed frame maximum length (e.g. 1518) it is interpreted as a type field.
Destination and source address	48-bit MAC addresses. The least significant byte is sent/received first and the first two least significant bits of the MAC address distinguish MAC frames, as detailed in MAC address check.

Note

Although the IEEE specification defines a maximum frame length, the MAC core provides the flexibility to program any value for the frame maximum length.

43.6.1.1 Pause Frames

The receiving device generates a pause frame to indicate a congestion to the emitting device, which should stop sending data.

Pause frames are indicated by the length/type set to 0x8808. The two first bytes of a pause frame following the type, defines a 16-bit opcode field set to 0x0001 always. A 16-bit pause quanta is defined in the frame payload bytes 2 (P1) and 3 (P2) as defined in the following table. The P1 pause quanta byte is the most significant.

Table 43-5. Pause Frame Format (Values in Hex)

1	2	3	4	5	6	7	8	9	10	11	12	13	14
55	55	55	55	55	55	55	D5	01	80	C2	00	00	01
Preamble							SFD	Multicast Destination Address					
15	16	17	18	19	20	21	22	23	24	25	26	27 –68	
00	00	00	00	00	00	88	08	00	01	hi	lo	00	
Source Address						Type		Opcode		P1	P2	pad (42)	
69	70	71	72										
26	6B	AE	0A										
CRC-32													

There is no payload length field found within a pause frame and a pause frame is always padded with 42 bytes (0x00).

If a pause frame with a pause value greater than zero (XOFF condition) is received, the MAC stops transmitting data as soon the current frame transfer is completed. The MAC stops transmitting data for the value defined in pause quanta. One pause quanta fraction refers to 512 bit times.

If a pause frame with a pause value of zero (XON condition) is received, the transmitter is allowed to send data immediately (see Full-duplex flow control operation for details).

43.6.1.2 Magic packets

A magic packet is a unicast, multicast, or broadcast packet, which carries a defined sequence in the payload section.

Magic packets are received and inspected only under specific conditions as described in Magic packet detection.

The defined sequence to decode a magic packet is formed with a synchronization stream which consists of six consecutive 0xFF bytes, and is followed by sequence of sixteen consecutive unicast MAC addresses of the node to be awakened.

This sequence can be located anywhere in the magic packet payload. The magic packet is formed with a standard Ethernet header, optional padding, and CRC.

43.6.2 IP and higher layers frame format

The following sections use the term datagram to describe the protocol specific data unit that is found within the payload section of its container entity.

For example, an IP datagram specifies the payload section of an Ethernet frame. A TCP datagram specifies the payload section within an IP datagram.

43.6.2.1 Ethernet types

IP datagrams are carried in the payload section of an Ethernet frame. The Ethernet frame type/length field discriminates several datagram types.

The following table lists the types of interest:

Table 43-6. Ethernet type value examples

Type	Description
0x8100	VLAN-tagged frame. The actual type is found 4 octets later in the frame.
0x0800	IPv4
0x0806	ARP
0x86DD	IPv6

43.6.2.2 IPv4 datagram format

The following figure shows the IP Version 4 (IPv4) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words. The first byte sent/received is the leftmost byte of the first word (in other words, version/IHL field).

The IP header can contain further options, which are always padded if necessary to guarantee the payload following the header is aligned to a 32-bit boundary.

The IP header is immediately followed by the payload, which can contain further protocol headers (for example, TCP or UDP, as indicated by the protocol field value). The complete IP datagram is transported in the payload section of an Ethernet frame.

Table 43-7. IPv4 header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Version	IHL	TOS		Length			
Fragment ID				Flags	Fragment offset		
TTL		Protocol		Header checksum			
Source address							
Destination address							
Options							

Table 43-8. IPv4 header fields

Field name	Description
Version	4-bit IP version information. 0x4 for IPv4 frames.
IHL	4-bit Internet header length information. Determines number of 32-bit words found within the IP header. If no options are present, the default value is 0x5.
TOS	Type of service/DiffServ field.
Length	Total length of the datagram in bytes, including all octets of header and payload.
Fragment ID, flags, fragment offset	Fields used for IP fragmentation.
TTL	Time-to-live. In effect, is decremented at each router arrival. If zero, datagram must be discarded.
Protocol	Identifier of protocol that follows in the datagram.
Header checksum	Checksum of IP header. For computational purposes, this field's value is zero.
Source address	Source IP address.
Destination address	Destination IP address.

43.6.2.3 IPv6 datagram format

The following figure shows the IP version 6 (IPv6) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words and has a fixed length of ten words (40 bytes). The next header field identifies the type of the header that follows the IPv6 header. It is defined similar to the protocol identifier within IPv4, with new definitions for identifying extension headers. These headers can be inserted between the IPv6 header and the protocol header, which will shift the protocol header accordingly.The accelerator currently only supports IPv6 without extension headers (in other words, the next header specifies TCP, UDP, or IMCP).

The first byte sent/received is the leftmost byte of the first word (in other words, version/ traffic class fields).

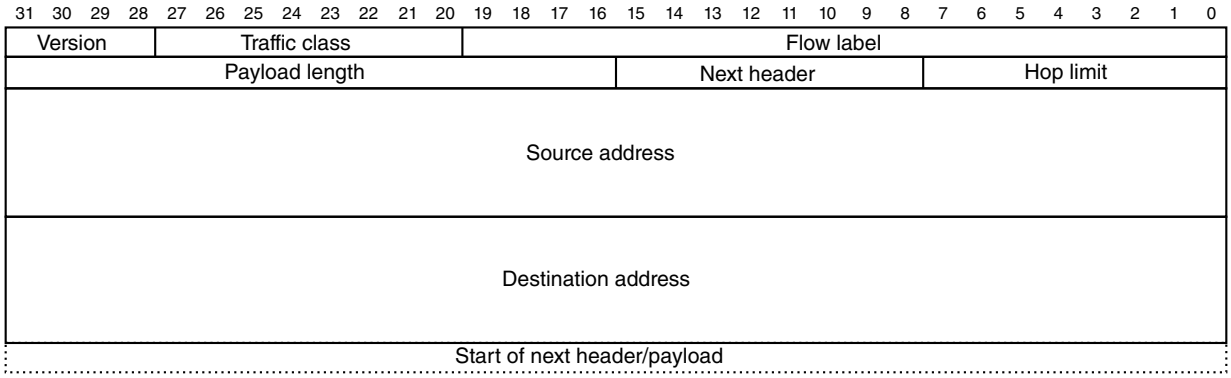


Figure 43-4. IPv6 header format

Table 43-9. IPv6 header fields

Field name	Description
Version	4-bit IP version information. 0x6 for all IPv6 frames.
Traffic class	8-bit field defining the traffic class.
Flow label	20-bit flow label identifying frames of the same flow.
Payload length	16-bit length of the datagram payload in bytes. It includes all octets following the IPv6 header.
Next header	Identifies the header that follows the IPv6 header. This can be the protocol header or any IPv6 defined extension header.
Hop limit	Hop counter, decremented by one by each station that forwards the frame. If hop limit is 0 the frame must be discarded.
Source address	128-bit IPv6 source address.
Destination address	128-bit IPv6 destination address.

43.6.2.4 Internet Control Message Protocol (ICMP) datagram format

An internet control message protocol (ICMP) is found following the IP header, if the protocol identifier is 1. The ICMP datagram has a four-octet header followed by additional message data.

Table 43-10. ICMP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Type		Code		Checksum			
ICMP message data							

Table 43-11. IP header fields

Field name	Description
Type	8-bit type information
Code	8-bit code that is related to the message type
Checksum	16-bit one's complement checksum over the complete ICMP datagram

43.6.2.5 User Datagram Protocol (UDP) datagram format

A user datagram protocol header is found after the IP header, when the protocol identifier is 17.

The payload of the datagram is after the UDP header. The header byte order follows the conventions given for the IP header above.

Table 43-12. UDP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source port				Destination port			
Length				Checksum			

Table 43-13. UDP header fields

Field name	Description
Source port	Source application port
Destination port	Destination application port
Length	Length of user data which immediately follows the header, including the UDP header (that is, minimum value is 8)
Checksum	Checksum over the complete datagram and some IP header information

43.6.2.6 TCP datagram format

A TCP header is found following the IP header, when the protocol identifier has a value of 6.

The TCP payload immediately follows the TCP header.

Table 43-14. TCP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source port				Destination port			

Table continues on the next page...

Table 43-14. TCP header format (continued)

Sequence number			
Acknowledgement number			
Offset	Reserved	Flags	Window
Checksum			Urgent pointer
Options			

Table 43-15. TCP header fields

Field name	Description
Source port	Source application port
Destination port	Destination application port
Sequence number	Transmit sequence number
Ack. number	Receive sequence number
Offset	Data offset, which is number of 32-bit words within TCP header — if no options selected, defaults to value of 5
Flags	URG, ACK, PSH, RST, SYN, FIN flags
Window	TCP receive window size information
Checksum	Checksum over the complete datagram (TCP header and data) and IP header information
Options	Additional 32-bit words for protocol options

43.6.3 IEEE 1588 message formats

The following sections describe the IEEE 1588 message formats.

43.6.3.1 Transport encapsulation

The precision time protocol (PTP) datagrams are encapsulated in Ethernet frames using the UDP/IP transport mechanism, or optionally, with the newer 1588v2 directly in Ethernet frames (layer 2).

Typically, multicast addresses are used to allow efficient distribution of the synchronization messages.

43.6.3.1.1 UDP/IP

The 1588 messages (v1 and v2) can be transported using UDP/IP multicast messages.

Table 43-16 shows IP multicast groups defined for PTP. The table also shows their respective MAC layer multicast address mapping according to RFC 1112 (last three octets of IP follow the fixed value of 01-00-5E).

Table 43-16. UDP/IP multicast domains

Name	IP Address	MAC Address mapping
DefaultPTPdomain	224.0.1.129	01-00-5E-00-01-81
AlternatePTPdomain1	224.0.1.130	01-00-5E-00-01-82
AlternatePTPdomain2	224.0.1.131	01-00-5E-00-01-83
AlternatePTPdomain3	224.0.1.132	01-00-5E-00-01-84

Table 43-17. UDP port numbers

Message type	UDP port	Note
Event	319	Used for SYNC and DELAY_REQUEST messages
General	320	All other messages (for example, follow-up, delay-response)

43.6.3.1.2 Native Ethernet (PTPv2)

In addition to using UDP/IP frames, IEEE 1588v2 defines a native Ethernet frame format that uses ethertype = 0x88F7. The payload of the Ethernet frame immediately contains the PTP datagram, starting with the PTPv2 header.

Besides others, version 2 adds a peer delay mechanism to allow delay measurements between individual point-to-point links along a path over multiple nodes. The following multicast domains are also defined in PTPv2.

Table 43-18. PTPv2 multicast domains

Name	MAC address
Normal messages	01-1B-19-00-00-00
Peer delay messages	01-80-C2-00-00-0E

43.6.3.2 PTP header

All PTP frames contain a common header that determines the protocol version and the type of message, which defines the remaining content of the message.

All multi-octet fields are transmitted in big-endian order (the most significant byte is transmitted/received first).

The last four bits of versionPTP are at the same position (second byte) for PTPv1 and PTPv2 headers. This allows accurate identification by inspecting the first two bytes of the message.

43.6.3.2.1 PTPv1 header

Table 43-19. Common PTPv1 message header

Offset	Octets	Bits							
		7	6	5	4	3	2	1	0
0	2	versionPTP = 0x0001							
2	2	versionNetwork							
4	16	subdomain							
20	1	messageType							
21	1	sourceCommunicationTechnology							
22	6	sourceUuid							
28	2	sourcePortId							
30	2	sequenceId							
32	1	control							
33	1	0x00							
34	2	flags							
36	4	reserved							

The type of message is encoded in the messageType and control fields as shown in Table 43-20 :

Table 43-20. PTPv1 message type identification

messageType	control	Message Name	Message
0x01	0x0	SYNC	Event message
0x01	0x1	DELAY_REQ	Event message
0x02	0x2	FOLLOW_UP	General message
0x02	0x3	DELAY_RESP	General message
0x02	0x4	MANAGEMENT	General message
other	other	—	Reserved

The field sequenceId is used to non-ambiguously identify a message.

43.6.3.2.2 PTPv2 header

Table 43-21. Common PTPv2 message header

Offset	Octets	Bits							
		7	6	5	4	3	2	1	0
0	1	transportSpecific				messageId			
1	1	reserved				versionPTP = 0x2			
2	2	messageLength							
4	1	domainNumber							
5	1	reserved							
6	2	flags							
8	8	correctionField							
16	4	reserved							
20	10	sourcePortIdentity							
30	2	sequenceId							
32	1	control							
33	1	logMeanMessageInterval							

The type of message is encoded in the field messageId as follows:

Table 43-22. PTPv2 message type identification

messageId	Message name	Message
0x0	SYNC	Event message
0x1	DELAY_REQ	Event message
0x2	PATH_DELAY_REQ	Event message
0x3	PATH_DELAY_RESP	Event message
0x4–0x7	—	Reserved
0x8	FOLLOW_UP	General message
0x9	DELAY_RESP	General message
0xa	PATH_DELAY_FOLLOW_UP	General message
0xb	ANNOUNCE	General message
0xc	SIGNALING	General message
0xd	MANAGEMENT	General message

The PTPv2 flags field contains further details on the type of message, especially if one-step or two-step implementations are used. The one- or two-step implementation is controlled by the TWO_STEP bit in the first octet of the flags field as shown below. Reserved bits are cleared.

Table 43-23. PTPv2 message flags field definitions

Bit	Name	Description
0	ALTERNATE_MASTER	See IEEE 1588 Clause 17.4
1	TWO_STEP	1 Two-step clock 0 One-step clock
2	UNICAST	1 Transport layer address uses a unicast destination address 0 Multicast is used
3	—	Reserved
4	—	Reserved
5	Profile specific	
6	Profile specific	
7	—	Reserved

43.6.4 MAC receive

The MAC receive engine performs the following tasks:

- Check frame framing
- Remove frame preamble and frame SFD field
- Discard frame based on frame destination address field
- Terminate pause frames
- Check frame length
- Remove payload padding if it exists
- Calculate and verify CRC-32
- Write received frames in the core receive FIFO

If the MAC is programmed to operate in half-duplex mode, it will also check if the frame is received with a collision.

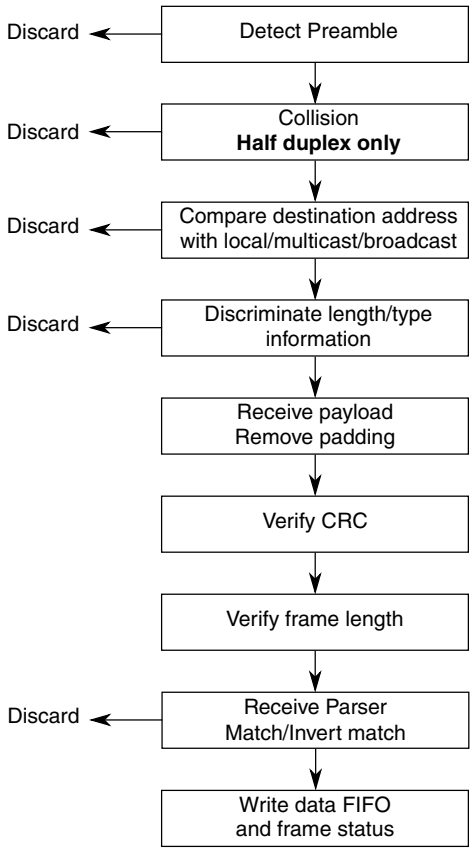


Figure 43-5. MAC receive flow

43.6.4.1 Collision detection in half-duplex mode

If the packet is received with a collision detected during reception of the first 64 bytes, the packet is discarded (if frame size was less than ~14 octets) or transmitted to the user application with an error and RxBD[CE] set.

43.6.4.2 Preamble processing

The IEEE 802.3 standard allows a maximum size of 56 bits (seven bytes) for the preamble, while the MAC core allows any preamble length, including zero length preamble.

The MAC core checks for the start frame delimiter (SFD) byte. If the next byte of the preamble, which is different from 0x55, is not 0xD5, the frame is discarded.

Although the IEEE specification dictates that the inner-packet gap should be at least 96 bits, the MAC core is designed to accept frames separated by only 64 10/100-Mbit/s operation (MII) bits.

The MAC core removes the preamble and SFD bytes.

43.6.4.3 MAC address check

The destination address bit 0 differentiates between multicast and unicast addresses.

- If bit 0 is 0, the MAC address is an individual (unicast) address.
- If bit 0 is 1, the MAC address defines a group (multicast) address.
- If all 48 bits of the MAC address are set, it indicates a broadcast address.

43.6.4.3.1 Unicast address check

If a unicast address is received, the destination MAC address is compared to the node MAC address programmed by the host in the PADDR1/2 registers.

If the destination address matches any of the programmed MAC addresses, the frame is accepted.

If Promiscuous mode is enabled (RCR[PROM] = 1) no address checking is performed and all unicast frames are accepted.

43.6.4.3.2 Multicast and unicast address resolution

The hash table algorithm used in the group and individual hash filtering operates as follows.

- The 48-bit destination address is mapped into one of 64 bits, represented by 64 bits in ENET*n*_GAUR/GALR (group address hash match) or ENET*n*_IAUR/IALR (individual address hash match).
- This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63.
- The msb of the CRC result selects ENET*n*_GAUR (msb = 1) or ENET*n*_GALR (msb = 0).
- The five lsbs of the hash result select the bit within the selected register.
- If the CRC generator selects a bit set in the hash table, the frame is accepted; else, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (or 87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The user must initialize the hash table registers. Use this CRC32 polynomial to compute the hash:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

If Promiscuous mode is enabled (ENET*n*_RCR[PROM] = 1) all unicast and multicast frames are accepted regardless of ENET*n*_GAUR/GALR and ENET*n*_IAUR/IALR settings.

43.6.4.3.3 Broadcast address reject

All broadcast frames are accepted if BC_REJ is cleared or ENET*n*_RCR[PROM] is set. If PROM is cleared when ENET*n*_RCR[BC_REJ] is set, all broadcast frames are rejected.

Table 43-24. Broadcast address reject programming

PROM	BC_REJ	Broadcast frames
0	0	Accepted
0	1	Rejected
1	0	Accepted
1	1	Accepted

43.6.4.3.4 Miss-bit implementation

For higher layer filtering purposes, RxBD[M] indicates an address miss when the MAC operates in promiscuous mode and accepts a frame that would otherwise be rejected.

If a group/individual hash or exact match does not occur and Promiscuous mode is enabled (RCR[PROM] = 1), the frame is accepted and the M bit is set in the buffer descriptor; otherwise, the frame is rejected.

This means the status bit is set in any of the following conditions during Promiscuous mode:

- A broadcast frame is received when BC_REJ is set

- A unicast is received that does not match either:
  - Node address (PALR[PADDR1] and PAUR[PADDR2])
  - Hash table for unicast (IAUR[IADDR1] and IALR[IADDR2])
- A multicast is received that does not match the GAUR[GADDR1] and GALR[GADDR2] hash table entries

43.6.4.4 Frame length/type verification: payload length check

If the length/type is less than 0x600 and NLC is set, the MAC checks the payload length and reports any error in the frame status word and interrupt bit PLR.

If the length/type is greater than or equal to 0x600, the MAC interprets the field as a type and no payload length check is performed.

The length check is performed on VLAN and stacked VLAN frames. If a padded frame is received, no length check can be performed due to the extended frame payload because padded frames can never have a payload length error.

43.6.4.5 Frame length/type verification: frame length check

When the receive frame length exceeds MAX_FL bytes, the BABR interrupt is generated and the RxBD[LG] bit is set.

The frame is not truncated unless the frame length exceeds the value programmed in ENET*n*_FTRL[TRUNC_FL]. If the frame is truncated, RxBD[TR] is set. In addition, a truncated frame always has the CRC error indication set (RxBD[CR]).

43.6.4.6 VLAN frames processing

VLAN frames have a length/type field set to 0x8100 immediately followed by a 16-Bit VLAN control information field.

VLAN-tagged frames are received as normal frames because the VLAN tag is not interpreted by the MAC function, and are pushed complete with the VLAN tag to the user application. If the length/type field of the VLAN-tagged frame, which is found four octets later in the frame, is less than 42, the padding is removed. In addition, the frame status word (RxBD[NO]) indicates that the current frame is VLAN tagged.

43.6.4.7 Pause frame termination

The receive engine terminates pause frames and does not transfer them to the receive FIFO. The quanta is extracted and sent to the MAC transmit path via a small internal clock rate decoupling asynchronous FIFO.

The quanta is written only if a correct CRC and frame length are detected by the control state machine. If not, the quanta is discarded and the MAC transmit path is not paused.

Good pause frames are ignored if ENET*n*_RCR[FCE] is cleared and are forwarded to the client interface when ENET*n*_RCR[PAUFWD] is set.

43.6.4.8 CRC check

The CRC-32 field is checked and forwarded to the core FIFO interface if ENET*n*_RCR[CRCFWD] is cleared and ENET*n*_RCR[PADEN] is set.

When CRCFWD is set (regardless of PADEN), the CRC-32 field is checked and terminated (not transmitted to the FIFO).

The CRC polynomial, as specified in the 802.3 standard, is:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

The 32 bits of the CRC value are placed in the frame check sequence (FCS) field with the x^{31} term as right-most bit of the first octet. The CRC bits are thus received in the following order: x^{31}, x^{30},..., x^{1}, x^{0}.

If a CRC error is detected, the frame is marked invalid and RxBD[CR] is set.

43.6.4.9 Frame padding removal

When a frame is received with a payload length field set to less than 46 (42 for VLAN-tagged frames and 38 for frames with stacked VLANs), the zero padding can be removed before the frame is written into the data FIFO depending on the setting of ENET*n*_RCR[PADEN].

Note

If a frame is received with excess padding (in other words, the length field is set as mentioned above, but the frame has more than 64 octets) and padding removal is enabled, then the

padding is removed as normal and no error is reported if the frame is otherwise correct (for example: good CRC, less than maximum length, and no other error).

43.6.4.10 Frame classification (AVB)

To support protocols such as Audio Video Bridging (AVB, IEEE 802.1Qav), normal traffic is separated from time-sensitive traffic immediately at the application interface to allow storing them in different queues for further processing. For every frame received, a classification based on VLAN priority can be performed. When a frame is received, and it contains a VLAN tag, the priority is compared against the values set in the classification match registers (see Receive Classification Match Register for Class n (ENET_RCMR*n*)) and the following figure.

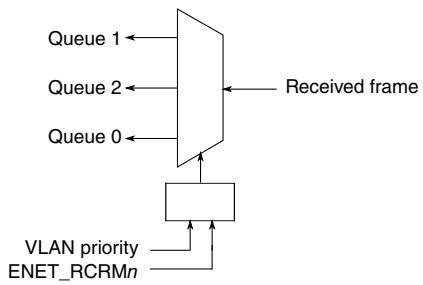


Figure 43-6. AVB frame classification

43.6.4.11 Frame parser

Ethernet frame pattern matching is used on receive frame to finally accept or reject the frame. When a frame is accepted, it is placed into a queue/ring depending on the matching entry configuration.

A matching entry table is implemented with a 96-bit pattern defined for each entry. A 96-bit pattern is separated into three 32-bit registers:

- MDATA: Data to be compared with the incoming frame. Data bits are used with the combination of mask bits.
- MMASK: Data bit within the MDATA will be used for comparison only when corresponding mask bit is set to 1.
- MCONFIG: Rule set for matching the data with incoming frame. See Match Entry Rules Configuration Register (ENET_MCONFIG) for more details.

The number of valid entries within the table can be defined through the RXPCTL register.

Initialize the entire parser memory to zero before creating entries in parser entry table.

Steps to write an entry within the matching entry table:

- Write the match registers MDATA, MMASK, and MCONFIG registers with desired configuration for the entry.
- Write the MENTRYRW register with the ENTRY_ADDRESS field as the table entry location and setting the WR field to 1.
- Wait until MENTRYRW[WR] is automatically cleared to 0 . When WR is cleared, write access is completed and table is ready for next read/write access.

Steps to read an entry from the matching entry table:

- Write the MENTRYRW register with the ENTRY_ADDRESS field as the table entry location and setting the RD field to 1.
- Wait until MENTRYRW[RD] is automatically cleared to 0. When RD is cleared, the 96-bit pattern is available and ready for reading from the registers MDATA, MMASK, and MCONFIG.
- Read the registers MDATA, MMASK, and MCONFIG.

Each pattern within the Match Entry table is compared sequentially starting from entry index0 with the incoming frame for match or invert match when the MCONFIG[AF] or MCONFIG[RF] field for that entry is non-zero. If match or invert match occurs for that entry and MCONFIG[AF] is set, the frame is accepted and placed into the queue defined by the MCONFIG[OK_INDEX] for that entry. The frame is rejected and EIR[PARSRF] is asserted when MCONFIG[RF] is set for that matching or invert-matching entry. See interrupt1 in following flowchart. When both MCONFIG[AF] and MCONFIG[RF] of the match entry is 0, the entry index of the table jumps to the index defined by MCONFIG[OK_INDEX] for the current entry.

When all entries are checked and no match or invert match is found or a parser programming error for that entry is found, EIR[PARSERR] is set to 1. See interrupt2 in the following flowchart.

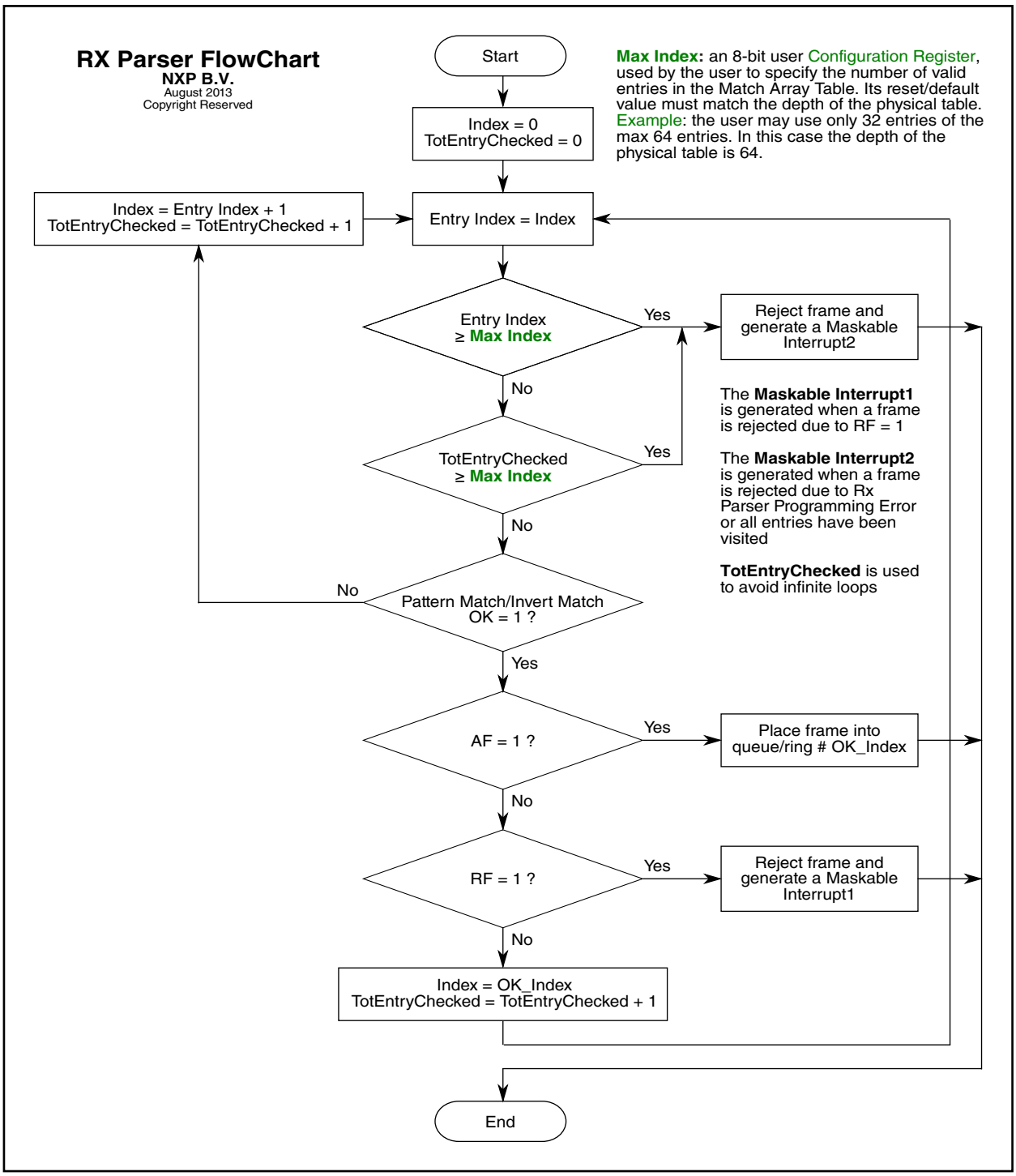


Figure 43-7. Receive parser flowchart

43.6.4.12 Receive flushing

RX flushing prevents frames in the RX FIFO from being blocked. Blocking can occur if the frame at the head of the RX FIFO cannot be forwarded because the ring it is associated with cannot accept it. This situation occurs when either the ring's RxBD[EMPTY] is not set or ENET_RDAR*n* is not set. When RX flushing is enabled, via QOS Scheme (ENET_QOS), the blocking frame will be flushed (discarded).

43.6.5 MAC transmit

Frame transmission starts when the transmit FIFO holds enough data.

After a transfer starts, the MAC transmit function performs the following tasks:

- Generates preamble and SFD field before frame transmission
- Generates XOFF pause frames if the receive FIFO reports a congestion or if ENET*n*_TCR[TFC_PAUSE] is set with ENET*n*_OPD[PAUSE_DUR] set to a non-zero value
- Generates XON pause frames if the receive FIFO congestion condition is cleared or if TFC_PAUSE is set with PAUSE_DUR cleared
- Suspends Ethernet frame transfer (XOFF) if a non-zero pause quanta is received from the MAC receive path
- Adds padding to the frame if required
- Calculates and appends CRC-32 to the transmitted frame
- Sends the frame with correct inter-packet gap (IPG) (deferring)

When the MAC is configured to operate in half-duplex mode, the following additional tasks are performed:

- Collision detection
- Frame retransmit after back-off timer expires

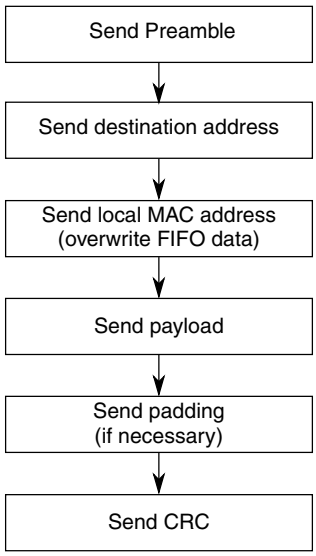


Figure 43-8. Frame transmit overview

43.6.5.1 Frame payload padding

The IEEE specification defines a minimum frame length of 64 bytes.

If the frame sent to the MAC from the user application has a size smaller than 60 bytes, the MAC automatically adds padding bytes (0x00) to comply with the Ethernet minimum frame length specification. Transmit padding is always performed and cannot be disabled.

If the MAC is not allowed to append a CRC (TxBD[TC] = 1), the user application is responsible for providing frames with a minimum length of 64 octets.

43.6.5.2 MAC address insertion

On each frame received from the core transmit FIFO interface, the source MAC address is either:

- Replaced by the address programmed in the PADDR1/2 fields (ENET*n*_TCR[ADDINS] = 1)
- Transparently forwarded to the Ethernet line (ENET*n*_TCR[ADDINS] = 0)

43.6.5.3 CRC-32 generation

The CRC-32 field is optionally generated and appended at the end of a frame.

The CRC polynomial, as specified in the 802.3 standard, is:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

The 32 bits of the CRC value are placed in the FCS field so that the x^{31} term is the right-most bit of the first octet. The CRC bits are thus transmitted in the following order: x^{31}, x^{30},..., x^{1}, x^{0}.

43.6.5.4 Inter-packet gap (IPG)

In full-duplex mode, after frame transmission and before transmission of a new frame, an IPG (programmed in ENET*n*_TIPG) is maintained. The minimum IPG can be programmed between 8 and 26 byte-times (64 and 208 bit-times).

In half-duplex mode, the core constantly monitors the line. Actual transmission of the data onto the network occurs only if it has been idle for a 96-bit time period, and any back-off time requirements have been satisfied. In accordance with the standard, the core begins to measure the IPG from CRS de-assertion.

43.6.5.5 Collision detection and handling — half-duplex operation only

A collision occurs on a half-duplex network when concurrent transmissions from two or more nodes take place. During transmission, the core monitors the line condition and detects a collision when the PHY device asserts COL.

When the core detects a collision while transmitting, it stops transmission of the data and transmits a 32-bit jam pattern. If the collision is detected during the preamble or the SFD transmission, the jam pattern is transmitted after completing the SFD, which results in a minimum 96-bit fragment. The jam pattern is a fixed pattern that is not compared to the actual frame CRC, and has a very low probability (0.532) of having a jam pattern identical to the CRC.

If a collision occurs before transmission of 64 bytes (including preamble and SFD), the MAC core waits for the backoff period and retransmits the packet data (stored in a 64-byte re-transmit buffer) that has already been sent on the line. The backoff period is generated from a pseudo-random process (truncated binary exponential backoff).

If a collision occurs after transmission of 64 bytes (including preamble and SFD), the MAC discards the remainder of the frame, optionally sets the LC interrupt bit, and sets TxBD[LCE].

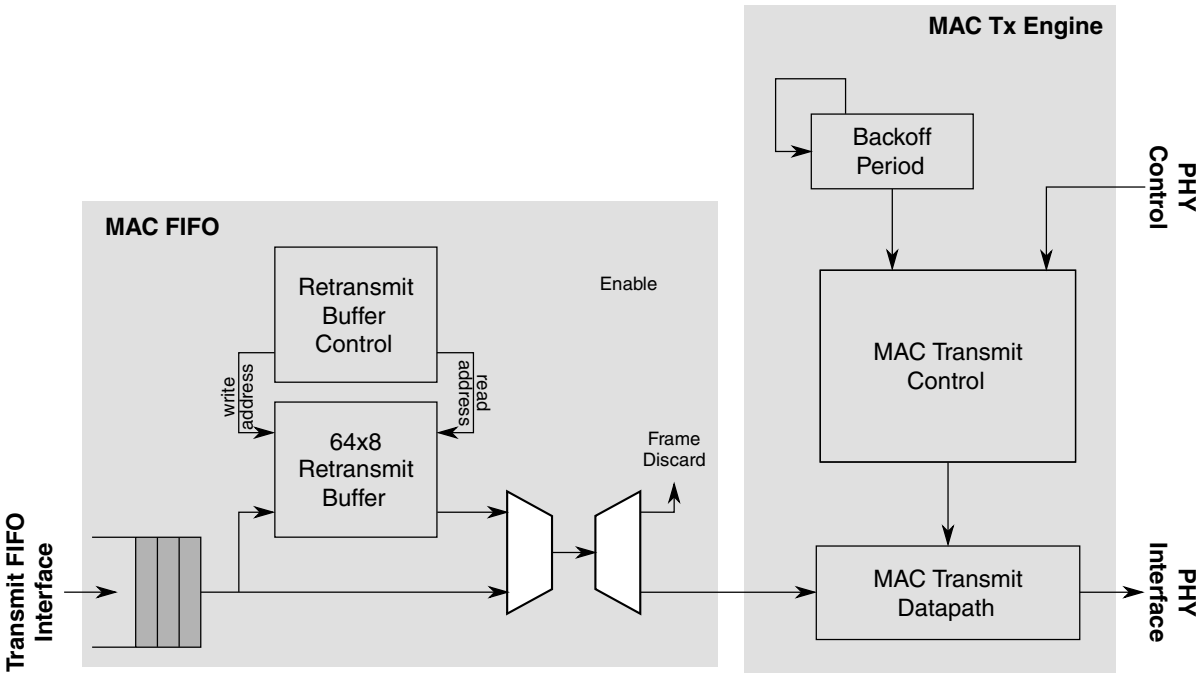


Figure 43-9. Packet re-transmit overview

The backoff time is represented by an integer multiple of slot times. One slot is equal to a 512-bit time period. The number of the delay slot times, before the n^{th} re-transmission attempt, is chosen as a uniformly-distributed random integer in the range:

- $0 < r < 2^k$
- $k = \min(n, N)$; where n is the number of retransmissions and $N = 10$

For example, after the first collision, the backoff period is 0 or 1 slot time. If a collision occurs on the first retransmission, the backoff period is 0, 1, 2, or 3, and so on.

The maximum backoff time (in 512-bit time slots) is limited by N = 10 as specified in the IEEE 802.3 standard.

If a collision occurs after 16 consecutive retransmissions, the core reports an excessive collision condition (ENET*n*_EIR[RL] interrupt field and TxBD[EE]) and discards the current packet from the FIFO.

In networks violating the standard requirements, a collision may occur after transmission of the first 64 bytes. In this case, the core stops the current packet transmission and discards the rest of the packet from the transmit FIFO. The core resumes transmission with the next packet available in the core transmit FIFO.

warning

Ethernet PHYs that support the SQE Test, or "heartbeat,"
feature must disable this feature. When this feature is enabled,

the PHY asserts the collision signal after a frame is transmitted to indicate to the ENET that the PHY's collision logic is working. This may cause data corruption in the next frame from the ENET. This corrupted frame contains up to 21 zero bytes which start somewhere within the MAC destination address field. The ENET, however, will still generate a good FCS (CRC-32) but with corrupted data.

43.6.5.6 Rate limiting / traffic shaping support

The MAC-NET supports two methods to optimize frame traffic for either time-sensitive AVB frames (Class A and Class B) or best-effort non-AVB frames:

- Round-robin scheme
- Credit-based traffic shaping

To ensure that sufficient bandwidth is allocated for the AVB frames, the credit-based shaper must be used. Either method can be combined with a time-based shaper to ensure that a frame is always transmitted in its correct time slot.

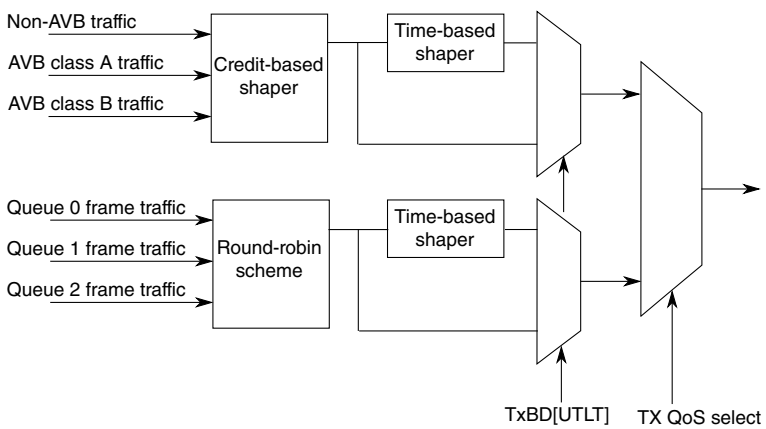


Figure 43-10. Transmit arbitration schemes

43.6.5.6.1 Round-robin policy

In the round-robin QoS scheme, each queue is given an equal opportunity to transmit one frame. For example, if queue *n* has a frame to transmit, the queue transmits its frame. After queue *n* has transmitted its frame, or if queue *n* does not have a frame to transmit, queue *n*+1 is then allowed to transmit its frame, and so on.

43.6.5.6.2 Credit-based shaper

The AVB credit based shaper acts independently, per class, to control the bandwidth distribution between normal traffic and time-sensitive traffic with respect to the total link bandwidth available. As per the IEEE 802.1Q, the maximum bandwidth distribution that can be allocated for the time-sensitive frames is 75%. The following example uses 70% for AVB frames and 30% for non-AVB frames:

Table 43-25. Bandwidth allocation example

AVB Class A	50%
AVB Class B	20%
non-AVB frames	30%
Total	100%

See DMA Class Based Configuration (ENET_DMA*n*CFG) for information on how to allocate bandwidth. The following figure shows how frame traffic is controlled within the MAC-NET core.

43.6.5.6.3 Time-based shaper

The time-based shaper enables the user to specify when a frame can be transmitted. It is always used in combination with either the round-robin scheme or the credit-based shaper. Use of the time-based shaper can ensure that frames are always transmitted in the correct time slot.

43.6.5.6.3.1 Time-based shaper example

Time-based shaping involves the following procedure:

1. Read the current value of the timer. See Timer Value Register (ENET_ATVR) and Adjustable Timer Control Register (ENET_ATCR).
2. Calculate the frame launch time and write this value to the TLT field of the frame's TxBD. See Enhanced transmit buffer descriptor.
3. Instruct the ENET to use the TLT by setting TxBD[UTLT].

The frame will be fetched and transmitted only if TxBD[TLT] is less than the current value of the timer. In other words, the timer must be past, that is, be greater than, the transmit launch time before the frame will be fetched and transmitted.

The transmit launch time must be not be greater than the current value of ENET_ATVR + (0.5 × ENET_ATPER). This means the application can not prepare frames with launch times beyond ENET_ATVR + (0.5 × ENET_ATPER). Because ENET_ATVR wraps, calculate TLT using the equation

$$(ENET_ATVR + (0.5 \times ENET_ATPER)) \bmod ENET_ATPER$$

For example, if ENET_ATPER is set to the recommended value of 1,000,000,000 ns (one second), then the user must not prepare future frames with launch times greater than (500,000,000 + ENET_ATVR).

As a specific example, if the current value of ENET_ATVR is 100,000, then you can prepare the following frames:

1. Frame 1 TxBD[TLT] = 100,000ns + 125,000ns
2. Frame 2 TxBD[TLT] = 100,000ns + 250,000ns
3. And so on

43.6.6 Full-duplex flow control operation

Three conditions are handled by the core's flow control engine:

- Remote device congestion — The remote device connected to the same Ethernet segment as the core reports congestion and requests that the core stop sending data.
- Core FIFO congestion — When the core's receive FIFO reaches a user-programmable threshold (RX section empty), the core sends a pause frame back to the remote device requesting the data transfer to stop.
- Local device congestion — Any device connected to the core can request (typically, via the host processor) the remote device to stop transmitting data.

43.6.6.1 Remote device congestion

When the MAC transmit control gets a valid pause quanta from the receive path and if ENET*n*_RCR[FCE] is set, the MAC transmit logic:

- Completes the transfer of the current frame.
- Stops sending data for the amount of time specified by the pause quanta in 512 bit time increments.
- Sets ENET*n*_TCR[RFC_PAUSE].

Frame transfer resumes when the time specified by the quanta expires and if no new quanta value is received, or if a new pause frame with a quanta value set to 0x0000 is received. The MAC also resets RFC_PAUSE to zero.

If ENET*n*_RCR[FCE] cleared, the MAC ignores received pause frames.

Optionally and independent of ENET*n*_RCR[FCE], pause frames are forwarded to the client interface if PAUFWD is set.

43.6.6.2 Local device/FIFO congestion

The MAC transmit engine generates pause frames when the local receive FIFO is not able to receive more than a pre-defined number of words (FIFO programmable threshold) or when pause frame generation is requested by the local host processor.

- To generate a pause frame, the host processor sets ENET*n*_TCR[TFC_PAUSE]. A single pause frame is generated when the current frame transfer is completed and TFC_PAUSE is automatically cleared. Optionally, an interrupt is generated.
- An XOFF pause frame is generated when the receive FIFO asserts its section empty flag (internal). An XOFF pause frame is generated automatically, when the current frame transfer completes.
- An XON pause frame is generated when the receive FIFO deasserts its section empty flag (internal). An XON pause frame is generated automatically, when the current frame transfer completes.

When an XOFF pause frame is generated, the pause quanta (payload byte P1 and P2) is filled with the value programmed in ENET*n*_OPD[PAUSE_DUR].

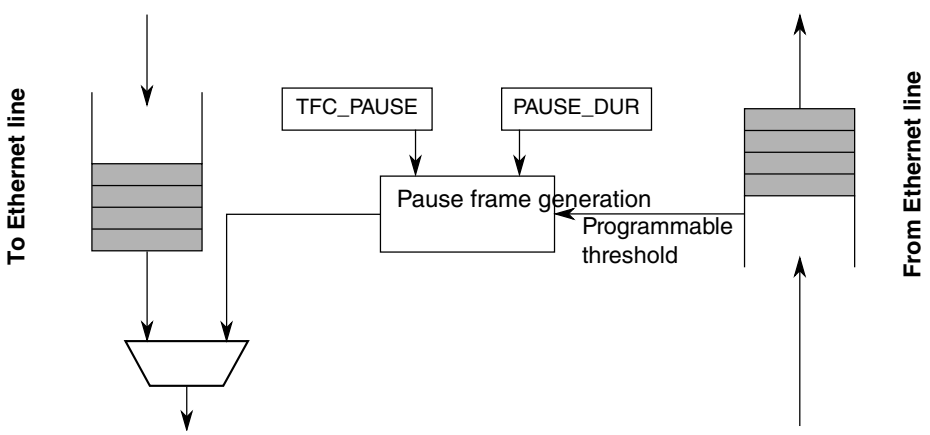


Figure 43-11. Pause frame generation overview

Note

Although the flow control mechanism should prevent any FIFO overflow on the MAC core receive path, the core receive FIFO is protected. When an overflow is detected on the receive FIFO,

the current frame is truncated with an error indication set in the frame status word. The frame should subsequently be discarded by the user application.

43.6.7 Magic packet detection

Magic packet detection wakes a node that is put in power-down mode by the node management agent. Magic packet detection is supported only if the MAC is configured in sleep mode.

43.6.7.1 Sleep mode

To put the MAC in Sleep mode, set ENET*n*_ECR[SLEEP]. At the same time ENET*n*_ECR[MAGICEN] should also be set to enable magic packet detection.

In addition, when the processor is in Stop mode, Sleep mode is entered, without affecting the ENET*n*_ECR register bits.

When the MAC is in Sleep mode:

- The transmit logic is disabled.
- The FIFO receive/transmit functions are disabled.
- The receive logic is kept in Normal mode, but it ignores all traffic from the line except magic packets. They are detected so that a remote agent can wake the node.

43.6.7.2 Magic packet detection

The core is designed to detect magic packets (see Magic packets) with the destination address set to:

- Any multicast address
- The broadcast address
- The unicast address programmed in PADDR1/2

When a magic packet is detected, EIR[WAKEUP] is set and none of the statistic registers are incremented.

43.6.7.3 Wakeup

When a magic packet is detected, indicated by ENET*n*_EIR[WAKEUP], ENET*n*_ECR[SLEEP] should be cleared to resume normal operation of the MAC. Clearing the SLEEP bit automatically masks ENET*n*_ECR[MAGICEN], disabling magic packet detection.

43.6.8 IP accelerator functions

The following sections describe the IP accelerator functions.

43.6.8.1 Checksum calculation

The IP and ICMP, TCP, UDP checksums are calculated with one's complement arithmetic summing up 16-bit values.

- For ICMP, the checksum is calculated over the complete ICMP datagram, in other words without IP header.
- For TCP and UDP, the checksums contain the header and data sections and values from the IP header, which can be seen as a pseudo-header that is not actually present in the data stream.

Table 43-26. IPv4 pseudo-header for checksum calculation

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source address							
Destination address							
Zero		Protocol		TCP/UDP length			

Table 43-27. IPv6 pseudo-header for checksum calculation

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source address							
Destination address							
TCP/UDP length							
Zero						Next header	

The TCP/UDP length value is the length of the TCP or UDP datagram, which is equal to the payload of an IP datagram. It is derived by subtracting the IP header length from the complete IP datagram length that is given in the IP header (IPv4), or directly taken from the IP header (IPv6). The protocol field is the corresponding value from the IP header. The Zero fields are all zeroes.

For IPv6, the complete 128-bit addresses are considered. The next header value identifies the upper layer protocol as either TCP or UDP. It may differ from the next header value of the IPv6 header if extension headers are inserted before the protocol header.

The checksum calculation uses 16-bit words in network byte order: The first byte sent/received is the MSB, and the second byte sent/received is the LSB of the 16-bit value to add to the checksum. If the frame ends on an odd number of bytes, a zero byte is appended for checksum calculation only, and is not actually transmitted.

43.6.8.2 Additional padding processing

According to IEEE 802.3, any Ethernet frame must have a minimum length of 64 octets.

The MAC usually removes padding on receive when a frame with length information is received. Because IP frames have a type value instead of length, the MAC does not remove padding for short IP frames, as it is not aware of the frame contents.

The IP accelerator function can be configured to remove the Ethernet padding bytes that might follow the IP datagram.

On transmit, the MAC automatically adds padding as necessary to fill any frame to a 64-byte length.

43.6.8.3 32-bit Ethernet payload alignment

The data FIFOs allow inserting two additional arbitrary bytes in front of a frame. This extends the 14-byte Ethernet header to a 16-byte header, which leads to alignment of the Ethernet payload, following the Ethernet header, on a 32-bit boundary.

This function can be enabled for transmit and receive independently with the corresponding SHIFT16 bits in the ENET*n*_TACC and ENET*n*_RACC registers.

When enabled, the valid frame data is arranged as shown in Table 43-28.

Table 43-28. 64-bit interface data structure with SHIFT16 enabled

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0	

Table continues on the next page...

Table 43-28. 64-bit interface data structure with SHIFT16 enabled (continued)

Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	Any value	Any value
Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8	Byte 7	Byte 6
...							

43.6.8.3.1 Receive processing

When ENET*n*_RACC[SHIFT16] is set, each frame is received with two additional bytes in front of the frame.

The user application must ignore these first two bytes and find the first byte of the frame in bits 23–16 of the first word from the RX FIFO.

Note

SHIFT16 must be set during initialization and kept set during the complete operation, because it influences the FIFO write behavior.

43.6.8.3.2 Transmit processing

When ENET*n*_TACC[SHIFT16] is set, the first two bytes of the first word written (bits 15–0) are discarded immediately by the FIFO write logic.

The SHIFT16 bit can be enabled/disabled for each frame individually if required, but can be changed only between frames.

43.6.8.4 Received frame discard

Because the receive FIFO must be operated in store and forward mode (ENET*n*_RSFL cleared), received frames can be discarded based on the following errors:

- The MAC function receives the frame with an error:
  - The frame has an invalid payload length
  - Frame length is greater than MAX_FL
  - Frame received with a CRC-32 error
  - Frame truncated due to receive FIFO overflow
  - Frame is corrupted as PHY signaled an error (RX_ERR asserted during reception)

- An IP frame is detected and the IP header checksum is wrong
- An IP frame with a valid IP header and a valid IP header checksum is detected, the protocol is known but the protocol-specific checksum is wrong

If one of the errors occurs and the IP accelerator function is configured to discard frames (ENET*n*_RACC), the frame is automatically discarded. Statistics are maintained normally and are not affected by this discard function.

43.6.8.5 IPv4 fragments

When an IPv4 IP fragment frame is received, only the IP header is inspected and its checksum verified. 32-bit alignment operates the same way on fragments as it does on normal IP frames, as specified above.

The IP fragment frame payload is not inspected for any protocol headers. As such, a protocol header would only exist in the very first fragment. To assist in protocol-specific checksum verification, the one's-complement sum is calculated on the IP payload (all bytes following the IP header) and provided with the frame status word.

The frame fragment status field (RxBD[FRAG]) is set to indicate a fragment reception, and the one's-complement sum of the IP payload is available in RxBD[Payload checksum].

Note

After all fragments have been received and reassembled, the application software can take advantage of the payload checksum delivered with the frame's status word to calculate the protocol-specific checksum of the datagram.

For example, if a TCP payload is delivered by multiple IP fragments, the application software can calculate the pseudo-header checksum value from the first fragment, and add the payload checksums delivered with the status for all fragments to verify the TCP datagram checksum.

43.6.8.6 IPv6 support

The following sections describe the IPv6 support.

43.6.8.6.1 Receive processing

An Ethernet frame of type 0x86DD identifies an IP Version 6 frame (IPv6) frame. If an IPv6 frame is received, the first IP header is inspected (first ten words), which is available in every IPv6 frame.

If the receive SHIFT16 function is enabled, the IP header is aligned on a 32-bit boundary allowing more efficient processing (see 32-bit Ethernet payload alignment).

For TCP and UDP datagrams, the pseudo-header checksum calculation is performed and verified.

To assist in protocol-specific checksum verification, the one's-complement sum is always calculated on the IP payload (all bytes following the IP header) and provided with the frame status word. For example, if extension headers were present, their sums can be subtracted in software from the checksum to isolate the TCP/UDP datagram checksum, if required.

43.6.8.6.2 Transmit processing

For IPv6 transmission, the SHIFT16 function is supported to process 32-bit aligned datagrams.

IPv6 has no IP header checksum; therefore, the IP checksum insertion configuration is ignored.

The protocol checksum is inserted only if the next header of the IP header is a known protocol (TCP, UDP, or ICMP). If a known protocol is detected, the checksum over all bytes following the IP header is calculated and inserted in the correct position.

The pseudo-header checksum calculation is performed for TCP and UDP datagrams accordingly.

43.6.9 Resets and stop controls

The following sections describe the resets and stop controls.

43.6.9.1 Hardware reset

To reset the Ethernet module, set ENET*n*_ECR[RESET].

43.6.9.2 Soft reset

When ENET*n*_ECR[ETHER_EN] is cleared during operation, the following occurs:

- uDMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers.
- A currently ongoing transmit is terminated by asserting TXER to the PHY.
- A currently ongoing transmit FIFO write from the application is terminated by stopping the write to the FIFO, and all further data from the application is ignored. All subsequent writes are ignored until re-enabled.
- A currently ongoing receive FIFO read is terminated. The RxBD has arbitrary values in this case.

43.6.9.3 Hardware freeze

When the processor enters debug mode and ECR[DBGEN] is set, the MAC enters a freeze state where it stops all transmit and receive activities gracefully.

The following happens when the MAC enters hardware freeze:

- A currently ongoing receive transaction on the receive application interface is completed as normal. No further frames are read from the FIFO.
- A currently ongoing transmit transaction on the transmit application interface is completed as normal (in other words, until writing end-of-packet (EOP)).
- A currently ongoing frame receive is completed normally, after which no further frames are accepted from the MII.
- A currently ongoing frame transmit is completed normally, after which no further frames are transmitted.

43.6.9.4 Graceful stop

During a graceful stop, any currently ongoing transactions are completed normally and no further frames are accepted. The MAC can resume from a graceful stop without the need for a reset (for example, clearing ETHER_EN is not required).

The following conditions lead to a graceful stop of the MAC transmit or receive datapaths.

43.6.9.4.1 Graceful transmit stop (GTS)

When gracefully stopped, the MAC is no longer reading frame data from the transmit FIFO and has completed any ongoing transmission.

In any of the following conditions, the transmit datapath stops after an ongoing frame transmission has been completed normally.

- ENET*n*_TCR[GTS] is set by software.
- ENET*n*_TCR[TFC_PAUSE] is set by software requesting a pause frame transmission. The status (and register bit) is cleared after the pause frame has been sent.
- A pause frame was received stopping the transmitter. The stopped situation is terminated when the pause timer expires or a pause frame with zero quanta is received.
- MAC is placed in Sleep mode by software or the processor entering Stop mode (see Sleep mode).
- The MAC is in Hardware Freeze mode.

When the transmitter has reached its stopped state, the following events occur:

- The GRA interrupt is asserted, when transitioned into stopped.
- In Hardware Freeze mode, the GRA interrupt does not wait for the application write completion and asserts when the transmit state machine (in other words, line side of TX FIFO) reaches its stopped state.

43.6.9.4.2 Graceful receive stop (GRS)

When gracefully stopped, the MAC is no longer writing frames into the receive FIFO.

The receive datapath stops after any ongoing frame reception has been completed normally, if any of the following conditions occur:

- MAC is placed in Sleep mode either by the software or the processor is in Stop mode). The MAC continues to receive frames and search for magic packets if enabled (see Magic packet detection). However, no frames are written into the receive FIFO, and therefore are not forwarded to the application.
- The MAC is in Hardware Freeze mode. The MAC does not accept any frames from the MII.

When the receive datapath is stopped, the following events occur:

- If the RX is in the stopped state, RCR[GRS] is set
- The GRA interrupt is asserted when the transmitter and receiver are stopped
- Any ongoing receive transaction to the application (RX FIFO read) continues normally until the frame end of package (EOP) is reached. After this, the following occurs:
  - When Sleep mode is active, all further frames are discarded, flushing the RX FIFO
  - In Hardware Freeze mode, no further frames are delivered to the application and they stay in the receive FIFO.

Note

The assertion of GRS does not wait for an ongoing FIFO read transaction on the application side of the FIFO (FIFO read).

43.6.9.4.3 Graceful stop interrupt (GRA)

The graceful stopped interrupt (GRA) is asserted for the following conditions:

- In Sleep mode, the interrupt asserts only after both TX and RX datapaths are stopped.
- In Hardware Freeze mode, the interrupt asserts only after both TX and RX datapaths are stopped.
- The MAC transmit datapath is stopped for any other condition (GTS, TFC_PAUSE, pause received).

The GRA interrupt is triggered only once when the stopped state is entered. If the interrupt is cleared while the stop condition persists, no further interrupt is triggered.

43.6.10 IEEE 1588 functions

To allow for IEEE 1588 or similar time synchronization protocol implementations, the MAC is combined with a time-stamping module to support precise time-stamping of incoming and outgoing frames. Set ENET*n*_ECR[EN1588] to enable 1588 support.

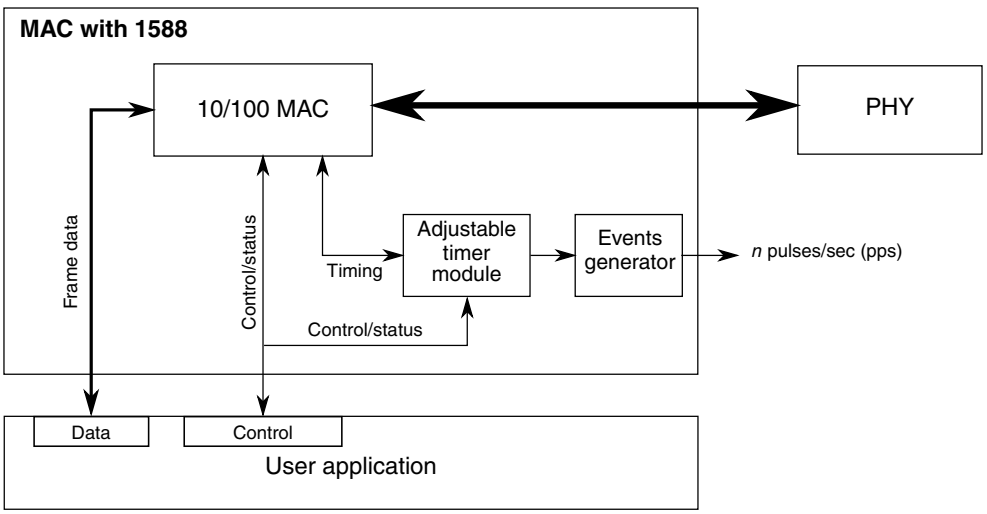


Figure 43-12. IEEE 1588 functions overview

43.6.10.1 Adjustable timer module

The adjustable timer module (TSM) implements the free-running counter (FRC), which generates the timestamps. The FRC operates with the time-stamping clock, which can be set to any value depending on your system requirements.

Through dedicated correction logic, the timer can be adjusted to allow synchronization to a remote master and provide a synchronized timing reference to the local system. The timer can be configured to cause an interrupt after a fixed time period, to allow synchronization of software timers or perform other synchronized system functions.

The timer is typically used to implement a period of one second; hence, its value ranges from 0 to (1×10^9)-1. The period event can trigger an interrupt, and software can maintain the seconds and hours time values as necessary.

43.6.10.1.1 Adjustable timer implementation

The adjustable timer consists of a programmable counter/accumulator and a correction counter. The periods of both counters and their increment rates are freely configurable, allowing very fine tuning of the timer.

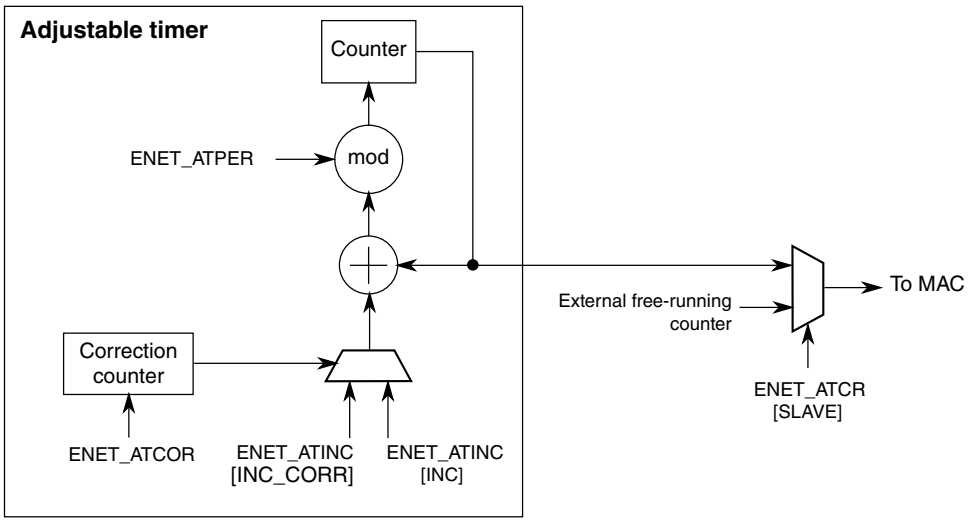


Figure 43-13. Adjustable timer implementation detail

The counter produces the current time. During each time-stamping clock cycle, a constant value is added to the current time as programmed in ENET*n*_ATINC. The value depends on the chosen time-stamping clock frequency. For example, if it operates at 125 MHz, setting the increment to eight represents 8 ns.

The period, configured in ENET*n*_ATPER, defines the modulo when the counter wraps. In a typical implementation, the period is set to 1×10^9 so that the counter wraps every second, and hence all timestamps represent the absolute nanoseconds within the one second period. When the period is reached, the counter wraps to start again respecting the period modulo. This means it does not necessarily start from zero, but instead the counter is loaded with the value (Current + Inc $-(1 \times 10^9)$), assuming the period is set to 1×10^9.

The correction counter operates fully independently, and increments by one with each time-stamping clock cycle. When it reaches the value configured in ENET*n*_ATCOR, it restarts and instructs the timer once to increment by the correction value, instead of the normal value.

The normal and correction increments are configured in ENET*n*_ATINC. To speed up the timer, set the correction increment more than the normal increment value. To slow down the timer, set the correction increment less than the normal increment value.

The correction counter only defines the distance of the corrective actions, not the amount. This allows very fine corrections and low jitter (in the range of 1 ns) independent of the chosen clock frequency.

By enabling slave mode (ENET*n*_ATCR[SLAVE] = 1), the timer is ignored and the current time is externally provided from one of the external modules. See the Chip Configuration details for which clock source is used. This is useful if multiple modules

within the system must operate from a single timer. When slave mode is enabled, you still must set ENET*n*_ATINC[INC] to the value of the master, since it is used for internal comparisons.

43.6.10.2 Transmit timestamping

Only 1588 event frames need to be time-stamped on transmit. The client application (for example, the MAC driver) should detect 1588 event frames and set TxBD[TS] together with the frame.

If TxBD[TS] is set, the MAC records the timestamp for the frame in ENET*n*_ATSTMP. ENET*n*_EIR[TS_AVAIL] is set to indicate that a new timestamp is available.

Software implements a handshaking procedure by setting TxBD[TS] when it transmits the frame for which a timestamp is needed, and then waits for ENET*n*_EIR[TS_AVAIL] to determine when the timestamp is available. The timestamp is then read from ENET*n*_ATSTMP. This is done for all event frames. Other frames do not use TxBD[TS] and, therefore, do not interfere with the timestamp capture.

43.6.10.3 Receive timestamping

When a frame is received, the MAC latches the value of the timer when the frame's start of frame delimiter (SFD) field is detected, and provides the captured timestamp on RxBD[1588 timestamp]. This is done for all received frames.

43.6.10.4 Time synchronization

The adjustable timer module is available to synchronize the local clock of a node to a remote master. It implements a free running 32-bit counter, and also contains an additional correction counter.

The correction counter increases or decreases the rate of the free running counter, enabling very fine granular changes of the timer for synchronization, yet adding only very low jitter when performing corrections.

The application software implements, in a slave scenario, the required control algorithm, setting the correction to compensate for local oscillator drifts and locking the timer to the remote master clock on the network.

The timer and all timestamp-related information should be configured to show the true nanoseconds value of a second (in other words, the timer is configured to have a period of one second). Hence, the values range from 0 to (1×10^9)–1. In this application, the seconds counter is implemented in software using an interrupt function that is executed when the nanoseconds counter wraps at 1×10^9.

43.6.10.5 Input Capture and Output Compare

The Input Capture Output Compare block can be used to provide precise hardware timing for input and output events.

43.6.10.5.1 Input capture

The TCCR*n* capture registers latch the time value when the corresponding external event occurs. An event can be a rising-, falling-, or either-edge of one of the 1588_TMR*n* signals. An event will cause the corresponding TCSR*n*[TF] timer flag to be set, indicating that an input capture has occurred. If the corresponding interrupt is enabled with the TCSR*n*[TIE] field, an interrupt can be generated.

43.6.10.5.2 Output compare

The TCCR*n* compare registers are loaded with the time at which the corresponding event should occur. When the ENET free-running counter value matches the output compare reference value in the TCCR*n* register, the corresponding flag, TCSR*n*[TF], is set, indicating that an output compare has occurred. The corresponding interrupt, if enabled by TCSR*n*[TIE], will be generated.The corresponding 1588_TMR*n* output signal will be asserted according to TCSR*n*[TMODE].

43.6.10.5.3 DMA requests

A DMA request can be enabled by setting TCSR*n*[TDRE]. The corresponding DMA request is generated when the TCSR*n*[TF] timer flag is set. When the DMA has completed, the corresponding TCSRn[TF] flag is cleared.

43.6.11 FIFO thresholds

The core FIFO thresholds are fully programmable to dynamically change the FIFO operation.

For example, store and forward transfer can be enabled by a simple change in the FIFO threshold registers.

The thresholds are defined in 64-bit words.

The receive and transmit FIFOs both have a depth of 256 words.

43.6.11.1 Receive FIFO

Four programmable thresholds are available, which can be set to any value to control the core operation as follows.

Table 43-29. Receive FIFO thresholds definition

Register	Description
ENET*n*_RSFL [RX_SECTION_FULL]	When the FIFO level reaches the ENET*n*_RSFL value, the MAC status signal is asserted to indicate that data is available in the receive FIFO (cut-through operation). Once asserted, if the FIFO empties below the threshold set with ENET*n*_RAEM and if the end-of-frame is not yet stored in the FIFO, the status signal is deasserted again. If a frame has a size smaller than the threshold (in other words, an end-of-frame is available for the frame), the status is also asserted. To enable store and forward on the receive path, clear ENET*n*_RSFL. The MAC status signal is asserted only when a complete frame is stored in the receive FIFO. When programming a non-zero value to ENET*n*_RSFL (cut-through operation) it should be greater than ENET*n*_RAEM.
ENET*n*_RAEM [RX_ALMOST_EMPTY]	When the FIFO level reaches the ENET*n*_RAEM value, and the end-of-frame has not been received, the core receive read control stops the FIFO read (and subsequently stops transferring data to the MAC client application). It continues to deliver the frame, if again more data than the threshold or the end-of-frame is available in the FIFO. Set ENET*n*_RAEM to a minimum of six.
ENET*n*_RAFL [RX_ALMOST_FULL]	When the FIFO level approaches the maximum and there is no more space remaining for at least ENETn_RAFL number of words, the MAC control logic stops writing data in the FIFO and truncates the receive frame to avoid FIFO overflow. The corresponding error status is set when the frame is delivered to the application. Set ENET*n*_RAFL to a minimum of 4.
ENET*n*_RSEM [RX_SECTION_EMPTY]	When the FIFO level reaches the ENET*n*_RSEM value, an indication is sent to the MAC transmit logic, which generates an XOFF pause frame. This indicates FIFO congestion to the remote Ethernet client. When the FIFO level goes below the value programmed in ENET*n*_RSEM, an indication is sent to the MAC transmit logic, which generates an XON pause frame. This indicates the FIFO congestion is cleared to the remote Ethernet client. Clearing ENET*n*_RSEM disables any pause frame generation.

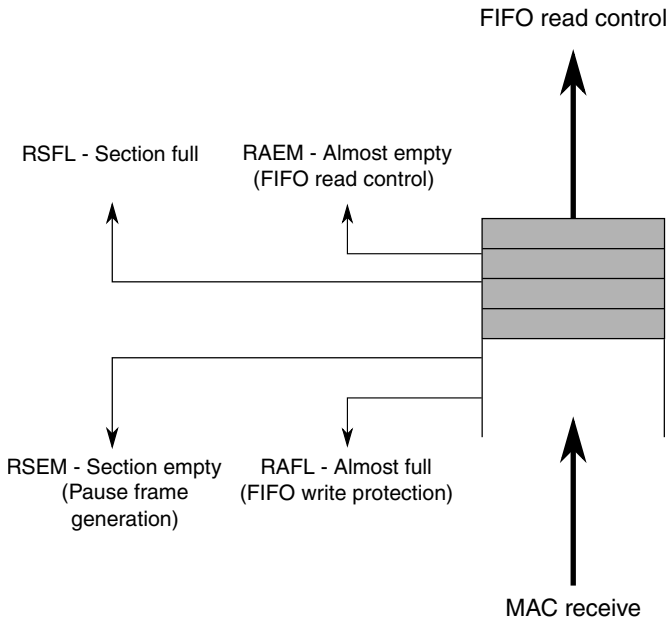


Figure 43-14. Receive FIFO overview

43.6.11.2 Transmit FIFO

Four programmable thresholds are available which control the core operation as described below.

Table 43-30. Transmit FIFO thresholds definition

Register	Description
ENET*n*_TAEM [TX_ALMOST _EMPTY]	When the FIFO level reaches the ENET*n*_TAEM value and no end-of-frame is available for the frame, the MAC transmit logic avoids a FIFO underflow by stopping FIFO reads and transmitting the Ethernet frame with an MII error indication. Set ENET*n*_TAEM to a minimum of 4.
ENET*n*_TAFL [TX_ALMOST _FULL]	When the FIFO level approaches the maximum, so that there is no more space for at least ENET*n*_TAFL number of words, the MAC deasserts its control signal to the application. If the application does not react on this signal, the FIFO write control logic avoids FIFO overflow by truncating the current frame and setting the error status. As a result, the frame is transmitted with an MII error indication. Set ENET*n*_TAFL to a minimum of 4. Larger values allow more latency for the application to react on the MAC control signal deassertion, before the frame is truncated. A typical setting is 8, which offers 3–4 clock cycles of latency to the application to react on the MAC control signal deassertion.
ENET*n*_TSEM [TX_SECTION _EMPTY]	When the FIFO level reaches the ENET*n*_TSEM value, a MAC status signal is deasserted to indicate that the transmit FIFO is getting full. This gives the ENET module an indication to slow or stop its write transaction to avoid a buffer overflow. This is a pure indication function to the application. It has no effect within the MAC. When ENET*n*_TSEM is 0, the signal is never deasserted.
ENET*n*_TFWR	When the FIFO level reaches the ENET*n*_TFWR value and when STRFWD is cleared, the MAC transmit control logic starts frame transmission before the end-of-frame is available in the FIFO (cut-through operation).

Table 43-30. Transmit FIFO thresholds definition

Register	Description
	If a complete frame has a size smaller than the ENET*n*_TFWR threshold, the MAC also transmits the frame to the line. To enable store and forward on the transmit path, set STRFWD. In this case, the MAC starts to transmit data only when a complete frame is stored in the transmit FIFO.

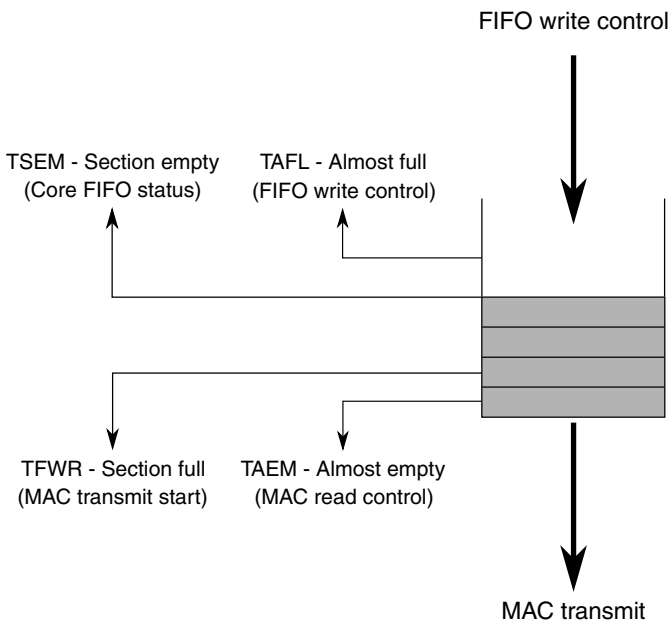


Figure 43-15. Transmit FIFO overview

43.6.12 Loopback options

The core implements external and internal loopback options, which are controlled by the ENET*n*_RCR register fields found here.

The core implements external and internal loopback options, which are controlled by the following ENET*n*_RCR register fields:

Table 43-31. Loopback options

Register field	Description
LOOP	Internal MII loopback. The MAC transmit is returned to the MAC receive. No data is transmitted to the external interfaces. In MII internal loopback, MII_TXCLK and MII_RXCLK must be provided with a clock signal (2.5 MHz for 10 Mbit/s, and 25 MHz for 100 Mbit/s))

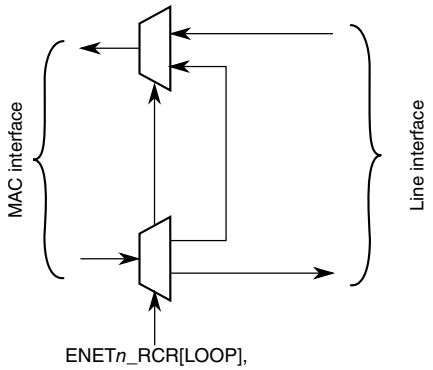


Figure 43-16. Loopback options

43.6.13 Legacy buffer descriptors

To support the Ethernet controller on previous chips, legacy FEC buffer descriptors are available. To enable legacy support, write 0 to ENET*n*_ECR[1588EN].

NOTE

- Legacy buffer descriptors are used in single-ring mode, that is, when DMA*n*CFG[DMA_CLASS_EN] are zero. In multi-ring mode, legacy TxBDs are used only with the round-robin scheme.
- The legacy buffer descriptor tables show the byte order for big-endian chips. DBSWP must be set to 0 after reset to enable big-endian mode.

43.6.13.1 Legacy receive buffer descriptor

The following table shows the legacy FEC receive buffer descriptor. Table 43-35 contains the descriptions for each field.

Table 43-32. Legacy FEC receive buffer descriptor (RxBD)

	Byte 0								Byte 1							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 2	Data length															
Offset + 4	Rx data buffer pointer — high halfword															
Offset + 6	Rx data buffer pointer — low halfword															

43.6.13.2 Legacy transmit buffer descriptor

The following table shows the legacy FEC transmit buffer descriptor. Table 43-37 contains the descriptions for each field.

Table 43-33. Legacy FEC transmit buffer descriptor (TxBD)

	Byte 0								Byte 1							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	R	TO1	W	TO2	L	TC	ABC[1]	—	—	—	—	—	—	—	—	—
Offset + 2	Data Length															
Offset + 4	Tx Data Buffer Pointer — high halfword															
Offset + 6	Tx Data Buffer Pointer — low halfword															

1. This field is not supported by the uDMA.

43.6.14 Enhanced buffer descriptors

This section provides a description of the enhanced operation of the driver/uDMA via the buffer descriptors.

It is followed by a detailed description of the receive and transmit descriptor fields. To enable the enhanced features, set ENET*n*_ECR[1588EN].

NOTE

The enhanced buffer descriptor tables show the byte order for big-endian chips. DBSWP must be set to 0 after reset to enable big-endian mode.

43.6.14.1 Enhanced receive buffer descriptor

The following table shows the enhanced uDMA receive buffer descriptor. Table 43-35 contains the descriptions for each field.

Table 43-34. Enhanced uDMA receive buffer descriptor (RxBD)

	Byte 0								Byte 1							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 2	Data length															
Offset + 4	Rx data buffer pointer – high halfword															

Table continues on the next page...

Table 43-34. Enhanced uDMA receive buffer descriptor (RxBD) (continued)

Offset + 6	Rx data buffer pointer – low halfword															
Offset + 8	ME	—	—	—	—	PE	CE	UC	INT	—	—	—	—	—	—	—
Offset + A	VPCP			—	—	—	—	—	—	—	ICE	PCR	—	VLAN	IPV6	FRAG
Offset + C	Header length					—	—	—	Protocol type							
Offset + E	Payload checksum															
Offset + 10	BDU	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 12	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 14	1588 timestamp – high halfword															
Offset + 16	1588 timestamp – low halfword															
Offset + 18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 43-35. Receive buffer descriptor field definitions

Word	Field	Description
Offset + 0	0 E	Empty. Written by the MAC (= 0) and user (= 1). 0 The data buffer associated with this BD is filled with received data, or data reception has aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 0	1 RO1	Receive software ownership. This field is reserved for use by software. This read/write field is not modified by hardware, nor does its value affect hardware.
Offset + 0	2 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in ENET*n*_RDSR.
Offset + 0	3 RO2	Receive software ownership. This field is reserved for use by software. This read/write field is not modified by hardware, nor does its value affect hardware.
Offset + 0	4 L	Last in frame. Written by the uDMA. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 0	5–6	Reserved, must be cleared.
Offset + 0	7 M	Miss. Written by the MAC. This field is set by the MAC for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, you can use this field to quickly determine whether the frame was destined to this station. This field is valid only if the L and PROM fields are set to 1. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode. The information needed for this field comes from the promiscuous_miss(ff_rx_err_stat[26]) sideband signal.
Offset + 0	8	Set if the DA is broadcast (FFFF_FFFF_FFFF).

Table continues on the next page...

Table 43-35. Receive buffer descriptor field definitions (continued)

Word	Field	Description
	BC	
Offset + 0	9 MC	Set if the DA is multicast and not BC.
Offset + 0	10 LG	Receive frame length violation. Written by the MAC. A frame length greater than RCR[MAX_FL] was recognized. This field is valid only if the L field is set. The receive data is not altered in any way unless the length exceeds TRUNC_FL bytes.
Offset + 0	11 NO	Receive non-octet aligned frame. Written by the MAC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error or a PHY error occurred. This field is valid only if the L field is set. If this field is set, the CR field is not set.
Offset + 0	12	Reserved, must be cleared.
Offset + 0	13 CR	Receive CRC or frame error. Written by the MAC. This frame contains a PHY or CRC error and is an integral number of octets in length. This field is valid only if the L field is set.
Offset + 0	14 OV	Overrun. Written by the MAC. A receive FIFO overrun occurred during frame reception. If this field is set, the other status fields, M, LG, NO, and CR, lose their normal meaning and are zero. This field is valid only if the L field is set.
Offset + 0	15 TR	Set if the receive frame is truncated (frame length >TRUNC_FL). If the TR field is set, the frame must be discarded and the other error fields must be ignored because they may be incorrect.
Offset + 2	0–15 Data Length	Data length. Written by the MAC. Data length is the number of octets written by the MAC into this BD's data buffer if L is cleared (the value is equal to EMRBR), or the length of the frame including CRC if L is set. It is written by the MAC once as the BD is closed.
0ffset + 4	0–15 Data buffer pointer high	Receive data buffer pointer, high halfword[1]
Offset + 6	0–15 Data buffer pointer low	Receive data buffer pointer, low halfword
Offset + 8	0 ME	MAC error. This field is written by the uDMA. This field means that the frame stored in the system memory was received with an error (typically, a receive FIFO overflow). This field is only valid when the L field is set.
Offset + 8	1–4	Reserved, must be cleared.
Offset + 8	5 PE	PHY Error. This field is written by the uDMA. Set to "1"when the frame was received with an Error character on the PHY interface. The frame is invalid. This field is valid only when the L field is set.
Offset + 8	6 CE	Collision. This field is written by the uDMA. Set when the frame was received with a collision detected during reception. The frame is invalid and sent to the user application. This field is valid only when the L field is set.
Offset + 8	7 UC	Unicast. This field is written by the uDMA, and means that the frame is unicast. This field is valid regardless of whether the L field is set.
Offset + 8	8 INT	Generate RXB/RXF interrupt. This field is set by the user to indicate that the uDMA is to generate an interrupt on the *dma_int_rxb* / *dma_int_rxfevent*.
Offset + 8	9–15	Reserved, must be cleared.

Table continues on the next page...

Table 43-35. Receive buffer descriptor field definitions (continued)

Word	Field	Description
Offset + A	0–2 VPCP	VLAN priority code point. This field is written by the uDMA to indicate the frame priority level. Valid values are from 0 (best effort) to 7 (highest). This value can be used to prioritize different classes of traffic (e.g., voice, video, data). This field is only valid if the L field is set.
Offset + A	3–9	Reserved, must be cleared.
Offset + A	10 ICE	IP header checksum error. This is an accelerator option. This field is written by the uDMA. Set when either a non-IP frame is received or the IP header checksum was invalid. An IP frame with less than 3 bytes of payload is considered to be an invalid IP frame. This field is only valid if the L field is set.
Offset + A	11 PCR	Protocol checksum error. This is an accelerator option. This field is written by the uDMA. Set when the checksum of the protocol is invalid or an unknown protocol is found and checksumming could not be performed. This field is only valid if the L field is set.
Offset + A	12	Reserved, must be cleared.
Offset + A	13 VLAN	VLAN. This is an accelerator option. This field is written by the uDMA. It means that the frame has a VLAN tag. This field is valid only if the L field is set.
Offset + A	14 IPV6	IPV6 Frame. This field is written by the uDMA. This field indicates that the frame has an IPv6 frame type. If this field is not set it means that an IPv4 or other protocol frame was received. This field is valid only if the L field is set.
Offset + A	15 FRAG	IPv4 Fragment.This is an accelerator option.This field is written by the uDMA. It indicates that the frame is an IPv4 fragment frame. This field is only valid when the L field is set.
Offset + C	0–4 Header length	Header length. This is an accelerator option. This field is written by the uDMA. This field is the sum of 32-bit words found within the IP and its following protocol headers. If an IP datagram with an unknown protocol is found, then the value is the length of the IP header. If no IP frame or an erroneous IP header is found, the value is 0. The following values are minimum values if no header options exist in the respective headers: • ICMP/IP: 6 (5 IP header, 1 ICMP header) • UDP/IP: 7 (5 IP header, 2 UDP header) • TCP/IP: 10 (5 IP header, 5 TCP header) This field is only valid if the L field is set.
Offset + C	5–7	Reserved, must be cleared.
Offset + C	8–15 Protocol type	Protocol type. This is an accelerator option. The 8-bit protocol field found within the IP header of the frame. It is valid only when ICE is cleared. This field is valid only when the L field is set.
Offset + E	0–15 Payload checksum	Internet payload checksum. This is an accelerator option. It is the one's complement sum of the payload section of the IP frame. The sum is calculated over all data following the IP header until the end of the IP payload. This field is valid only when the L field is set.
Offset + 10	0 BDU	Last buffer descriptor update done. Indicates that the last BD data has been updated by uDMA. This field is written by the user (=0) and uDMA (=1).
Offset + 10	1–15	Reserved, must be cleared.
Offset + 12	0–15	Reserved, must be cleared.
Offset + 14	0–15 1588 timestamp	This value is written by the uDMA. It is only valid if the L field is set.
Offset + 16		
Offset + 18	0–15	Reserved, must be cleared.

Table 43-35. Receive buffer descriptor field definitions

Word	Field	Description
– Offset + 1E		

1. The receive buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 64. The buffer must reside in memory external to the MAC. The Ethernet controller never modifies this value.

43.6.14.2 Enhanced transmit buffer descriptor

Table 43-36. Enhanced transmit buffer descriptor (TxBD)

	Byte 0								Byte 1							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Offset + 0	R	TO1	W	TO2	L	TC	—	—	—	—	—	—	—	—	—	—
Offset + 2	Data length															
Offset + 4	Tx Data Buffer Pointer – high halfword															
Offset + 6	Tx Data Buffer Pointer – low halfword															
Offset + 8	—	INT	TS	PINS	IINS	—	—	UTLT	FTYPE				—	—	—	—
Offset + A	TXE	—	UE	EE	FE	LCE	OE	TSE	—	—	—	—	—	—	—	—
Offset + C	TLT – high halfword															
Offset + E	TLT – low halfword															
Offset + 10	BDU	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 12	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 14	1588 timestamp – high halfword															
Offset + 16	1588 timestamp – low halfword															
Offset + 18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 43-37. Enhanced transmit buffer descriptor field definitions

Word	Field	Description
Offset + 0	0 R	Ready. Written by the MAC and user. 0 The data buffer associated with this BD is not ready for transmission. You are free to manipulate this BD or its associated data buffer. The MAC clears this field after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, prepared for transmission by you, has not been transmitted or currently transmits. You may write no fields of this BD after this field is set.
Offset + 0	1	Transmit software ownership. This field is reserved for software use. This read/write field is not modified by hardware and its value does not affect hardware.

Table continues on the next page...

Table 43-37. Enhanced transmit buffer descriptor field definitions (continued)

Word	Field	Description
	TO1	
Offset + 0	2 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ETDSR.
Offset + 0	3 TO2	Transmit software ownership. This field is reserved for use by software. This read/write field is not modified by hardware and its value does not affect hardware.
Offset + 0	4 L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame 1 The buffer is the last in the transmit frame
Offset + 0	5 TC	Transmit CRC. Written by user, and valid only when L is set. 0 End transmission immediately after the last data byte 1 Transmit the CRC sequence after the last data byte This field is valid only when the L field is set.
Offset + 0	6 ABC	Append bad CRC. **Note:** This field is not supported by the uDMA and is ignored.
Offset + 0	7–15	Reserved, must be cleared.
Offset + 2	0–15 Data Length	Data length, written by user. Data length is the number of octets the MAC should transmit from this BD's data buffer. It is never modified by the MAC.
Offset + 4	0–15 Data buffer pointer high	Tx data buffer pointer, high halfword. The buffer must reside in memory external to the MAC. This value is never modified by the Ethernet controller. **NOTE:** For optimal performance, make the transmit buffer pointer evenly divisible by 64.
Offset + 6	0–15 Data buffer pointer low	Tx data buffer pointer, low halfword
Offset + 8	0	Reserved, must be cleared.
Offset + 8	1 INT	Generate interrupt flags. This field is written by the user. This field is valid regardless of the L field and must be the same for all EBD for a given frame. The uDMA does not update this value.
Offset + 8	2 TS	Timestamp. This field is written by the user. This indicates that the uDMA is to generate a timestamp frame to the MAC. This field is valid regardless of the L field and must be the same for all EBD for the given frame. The uDMA does not update this value.
Offset + 8	3 PINS	Insert protocol specific checksum. This field is written by the user. If set, the MAC's IP accelerator calculates the protocol checksum and overwrites the corresponding checksum field with the calculated value. The checksum field must be cleared by the application generating the frame. The uDMA does not update this value. This field is valid regardless of the L field and must be the same for all EBD for a given frame.
Offset + 8	4 IINS	Insert IP header checksum. This field is written by the user. If set, the MAC's IP accelerator calculates the IP header checksum and overwrites the corresponding header field with the calculated value. The checksum field must be cleared by

Table continues on the next page...

Table 43-37. Enhanced transmit buffer descriptor field definitions (continued)

Word	Field	Description
		the application generating the frame. The uDMA does not update this value. This field is valid regardless of the L field and must be the same for all EBD for a given frame.
Offset + 8	5-6	Reserved, must be cleared.
Offset + 8	7 UTLT	Use transmit launch time. If set to 1, TLT high and low values are used to determine if the frame can be transmitted. This field must only be set in the *first* BD of a frame. TLT is always used in combination with either the round-robin scheme or the credit-based shaper. TLT can be used with single a BD queue. However, at least one DMA class, DMA*n*CFG[DMA_CLASS_ EN] must be enabled. For example, to use TLT with only queue 0, enable one of the DMA*n*CFG[DMA_CLASS_EN] fields and clear both TDAR1 and TDAR2. Although this enables multi-queue, only the single queue 0 is used.
Offset + 8	8–11 FTYPE	Type of frame to be transmitted. If a credit-based scheme is used, this field must match the BD ring queue. 0x0 – Non-AVB. Corresponds to ENET_TDSR. 0x1 – AVB Class A. Corresponds to ENET_TDSR1. 0x2 – AVB Class B. Corresponds to ENET_TDSR2. All other values are reserved.
Offset + 8	12–15	Reserved, must be cleared.
Offset + A	0 TXE	Transmit error occurred. This field is written by the uDMA. This field indicates that there was a transmit error of some sort reported with the frame. Effectively this field is an OR of the other error fields including UE, EE, FE, LCE, OE, and TSE. This field is valid only when the L field is set.
Offset + A	1	Reserved, must be cleared.
Offset + A	2 UE	Underflow error. This field is written by the uDMA. This field indicates that the MAC reported an underflow error on transmit. This field is valid only when the L field is set.
Offset + A	3 EE	Excess Collision error. This field is written by the uDMA. This field indicates that the MAC reported an excess collision error on transmit. This field is valid only when the L field is set.
Offset + A	4 FE	Frame with error. This field is written by the uDMA. This field indicates that the MAC reported that the uDMA reported an error when providing the packet. This field is valid only when the L field is set.
Offset + A	5 LCE	Late collision error. This field is written by the uDMA. This field indicates that the MAC reported that there was a Late Collision on transmit. This field is valid only when the L field is set.
Offset + A	6 OE	Overflow error. This field is written by the uDMA. This field indicates that the MAC reported that there was a FIFO overflow condition on transmit. This field is only valid when the L field is set.
Offset + A	7 TSE	Timestamp error. This field is written by the uDMA. This field indicates that the MAC reported a different frame type then a timestamp frame. This field is valid only when the L field is set.
Offset + A	8–15	Reserved, must be cleared.
Offset + C	0–15 TLT high	Transmit launch time high. Specifies when frame can by transmitted.

Table continues on the next page...

Table 43-37. Enhanced transmit buffer descriptor field definitions (continued)

Word	Field	Description
Offset + E	0–15 TLT low	Transmit launch time low. Specifies when frame can by transmitted.
Offset + 10	0 BDU	Last buffer descriptor update done. Indicates that the last BD data has been updated by uDMA. This field is written by the user (=0) and uDMA (=1).
Offset + 10	1–15	Reserved, must be cleared.
Offset + 12	0–15	Reserved, must be cleared.
Offset + 14	0–15	This value is written by the uDMA . It is valid only when the L field is set.
Offset + 16	1588 timestamp	
Offset + 18–Offset + 1E	0–15	Reserved, must be cleared.

43.6.15 Client FIFO application interface

The FIFO interface is completely asynchronous from the Ethernet line, and the transmit and receive interface can operate at a different clock rate.

All transfers to/from the user application are handled independently of the core operation, and the core provides a simple interface to user applications based on a two-signal handshake.

43.6.15.1 Data structure description

The data structure defined in the following tables for the FIFO interface must be respected to ensure proper data transmission on the Ethernet line. Byte 0 is sent to and received from the line first.

Table 43-38. FIFO interface data structure

	63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Word 0	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Word 1	Byte 15	Byte 14	Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8
...	...							

The size of a frame on the FIFO interface may not be a modulo of 64-bit.

The user application may not care about the Ethernet frame formats in full detail. It needs to provide and receive an Ethernet frame with the following structure:

- Ethernet MAC destination address

- Ethernet MAC source address
- Optional 802.1q VLAN tag (VLAN type and info field)
- Ethernet length/type field
- Payload

Frames on the FIFO interface do not contain preamble and SFD fields, which are inserted and discarded by the MAC on transmit and receive, respectively.

- On receive, CRC and frame padding can be stripped or passed through transparently.
- On transmit, padding and CRC can be provided by the user application, or appended automatically by the MAC independently for each frame. No size restrictions apply.

Note

On transmit, if ENET*n*_TCR[ADDINS] is set, bytes 6–11 of each frame can be set to any value, since the MAC overwrites the bytes with the MAC address programmed in the ENET*n*_PAUR and ENET*n*_PALR registers.

Table 43-39. FIFO interface frame format

Byte number	Field
0–5	Destination MAC address
6–11	Source MAC address
12–13	Length/type field
14–N	Payload data

VLAN-tagged frames are supported on both transmit and receive, and implement additional information (VLAN type and info).

Table 43-40. FIFO interface VLAN frame format

Byte number	Field
0–5	Destination MAC address
6–11	Source MAC address
12–15	VLAN tag and info
16–17	Length/type field
18–N	Payload data

Note

The standard defines that the LSB of the MAC address is sent/received first, while for all the other header fields — in other words, length/type, VLAN tag, VLAN info, and pause quanta — the MSB is sent/received first.

43.6.15.2 Data structure examples

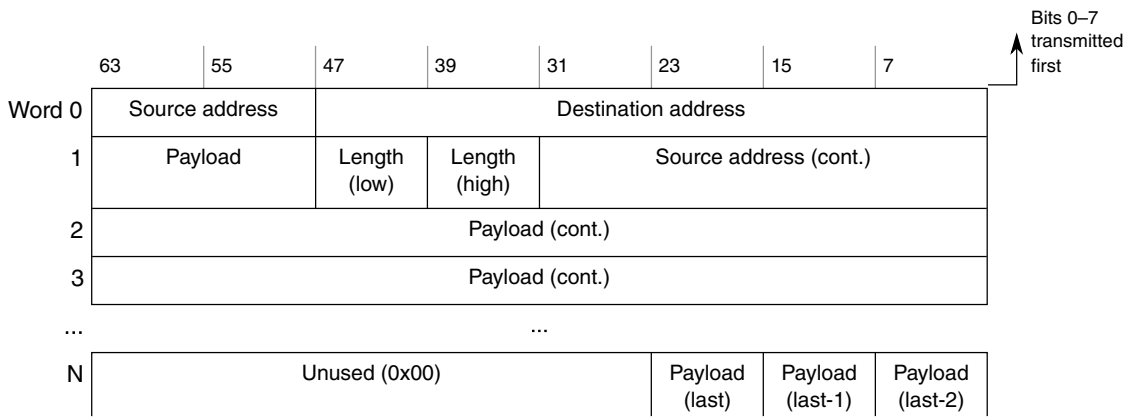


Figure 43-17. Normal Ethernet frame 64-bit mapping example

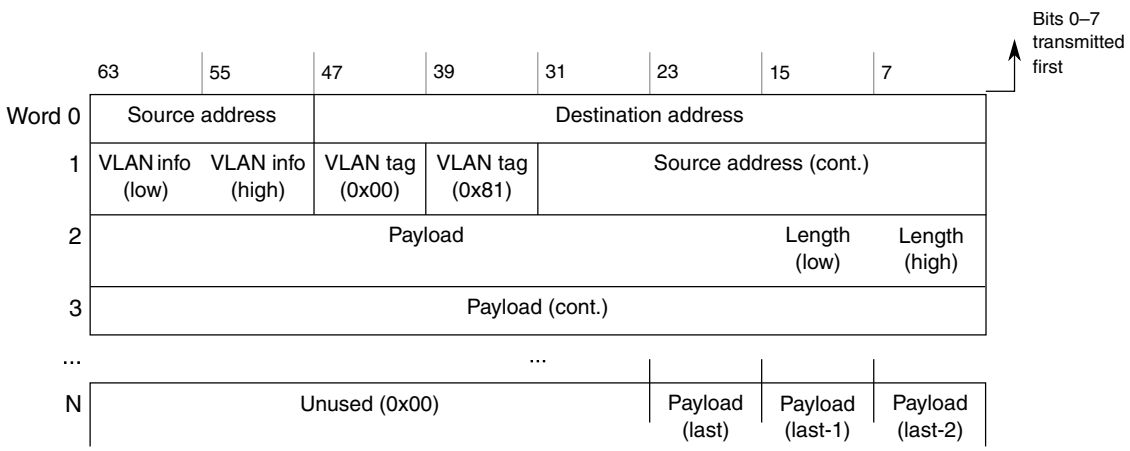


Figure 43-18. VLAN-tagged frame 64-bit mapping example

If CRC forwarding is enabled (CRCFWD = 0), the last four valid octets of the frame contain the FCS field. The non-significant bytes of the last word can have any value.

43.6.15.3 Frame status

A MAC layer status word and an accelerator status word is available in the receive buffer descriptor.

See Enhanced buffer descriptors for details.

The status is available with each frame with the last data of the frame.

If the frame status contains a MAC layer error (for example, CRC or length error), RxBD[ME] is also set with the last data of the frame.

43.6.16 FIFO protection

The following sections describe the FIFO protection mechanisms.

43.6.16.1 Transmit FIFO underflow

During a frame transfer, when the transmit FIFO reaches the almost empty threshold with no end-of-frame indication stored in the FIFO, the MAC logic:

- Stops reading data from the FIFO
- Asserts the MII error signal (MII_TXER) (1 in Figure 43-19) to indicate that the fragment already transferred is not valid
- Deasserts the MII transmit enable signal (MII_TXEN) to terminate the frame transfer (2)

After an underflow, when the application completes the frame transfer (3), the MAC transmit logic discards any new data available in the FIFO until the end of packet is reached (4) and sets the enhanced TxBD[UE] field.

The MAC starts to transfer data on the MII interface when the application sends a new frame with a start of frame indication (5).

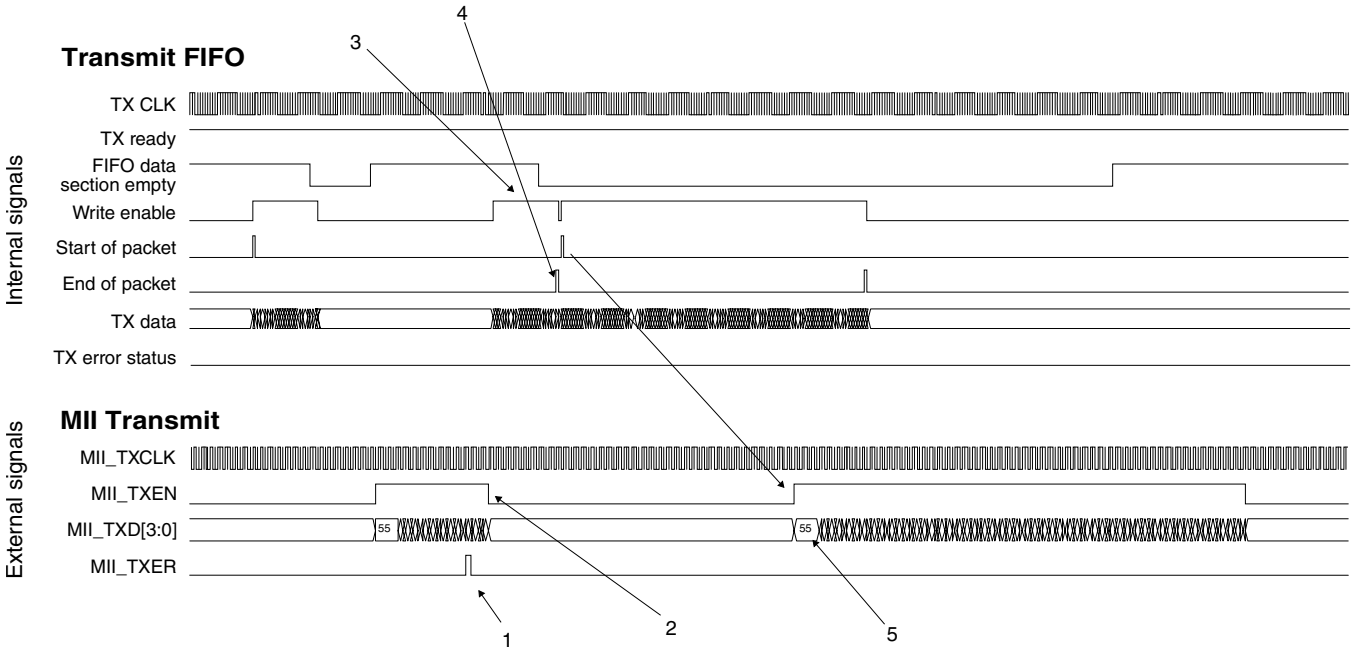


Figure 43-19. Transmit FIFO underflow protection

43.6.16.2 Transmit FIFO overflow

On the transmit path, when the FIFO reaches the programmable almost full threshold, the internal MAC ready signal is deasserted. The application should stop sending new data.

However, if the application keeps sending data, the transmit FIFO overflows, corrupting contents that were previously stored. The core logic sets the enhanced TxBD[OE] field for the next frame transmitted to indicate this overflow occurence.

Note

Overflow is a fatal error and must be addressed by resetting the core or clearing ENET*n*_ECR[ETHER_EN], to clear the FIFOs and prepare for normal operation again.

43.6.16.3 Receive FIFO overflow

During a frame reception, if the client application is not able to receive data (1), the MAC receive control truncates the incoming frame when the FIFO reaches the programmable almost-full threshold to avoid an overflow.

The frame is subsequently received on the FIFO interface with an error indication (enhanced RxBD[ME] field set together with receive end-of-packet) (2) with the truncation error status field set (3).

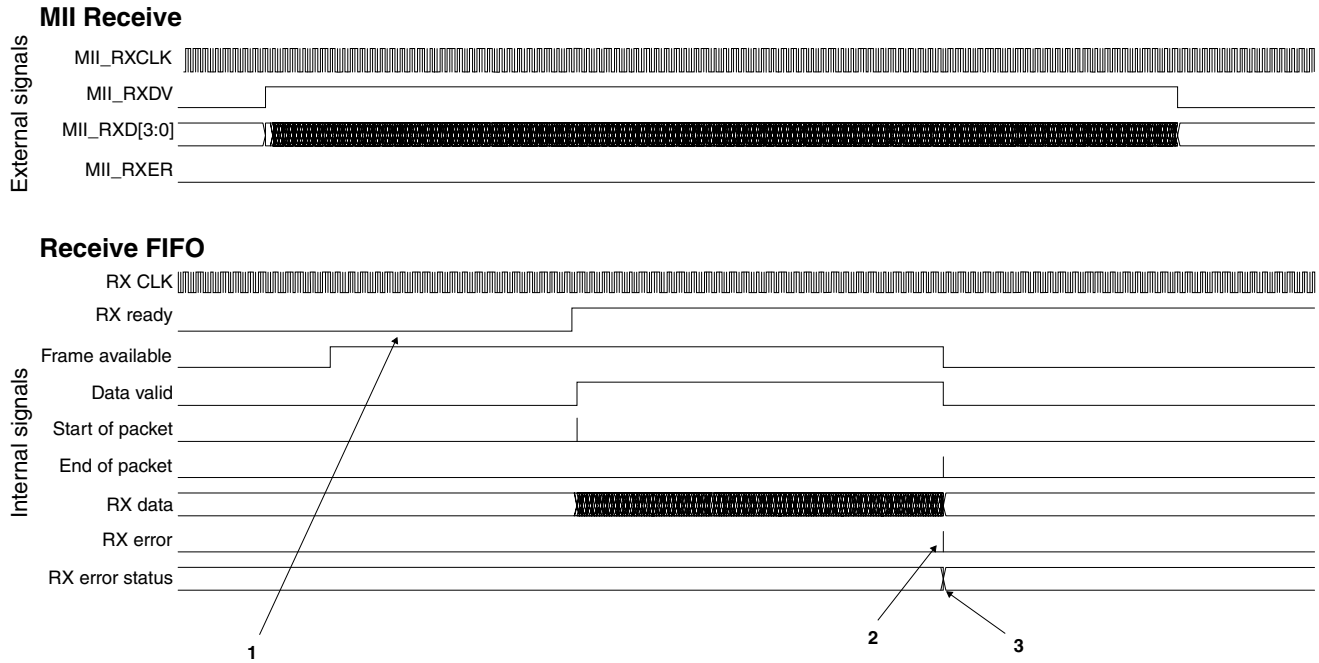


Figure 43-20. Receive FIFO overflow protection

43.6.17 Reference clock

The input clocks to the ENET module must meet the specifications in the following table.

Ethernet speed mode	Ethernet bus clock	Minimum ENET system clock
10 Mbit/s	2.5 MHz	5 MHz
100 Mbit/s	25.0 MHz	50 MHz

43.6.18 PHY management interface

The MDIO interface is a two-wire management interface. The MDIO management interface implements a standardized method to access the PHY device management registers.

The core implements a master MDIO interface, which can be connected to up to 32 PHY devices.

43.6.18.1 MDIO clause 22 frame format

The core MDIO master controller communicates with the slave (PHY device) using frames that are defined in the following table.

A complete frame has a length of 64 bits made up of an optional 32-bit preamble, 14-bit command, 2-bit bus direction change, and 16-bit data. Each bit is transferred on the rising edge of the MDIO clock (MDC signal). The MDIO data signal is tri-stated between frames.

The core PHY management interface supports the standard MDIO specification (IEEE 802.3 Clause 22).

Table 43-41. MDIO clause 22 frame structure

ST	OP	PHYADR	REGADR	TA	DATA

Table 43-42. MDIO frame field descriptions

Field	Description
ST (2 bits)	Start indication field, programmed with ENET*n*_MMFR[ST] and equal to 01 for Standard MDIO (Clause 22).
OP (2 bits)	Opcode defines type of operation. Programmed with ENET*n*_MMFR[OP]. 01 Write operation 10 Read operation
PHYADR (5 bits)	Five-bit PHY device address, programmed with ENET*n*_MMFR[PA]. Up to 32 devices can be addressed.
REGADR (5 bits)	Five-bit register address, programmed with ENET*n*_MMFR[RA]. Each PHY can implement up to 32 registers.
TA (2 bits)	Turnaround time, programmed with ENET*n*_MMFR[TA]. Two bit-times are reserved for read operations to switch the data bus from write to read. The PHY device presents its register contents in the data phase and drives the bus from the second bit of the turnaround phase.
Data (16 bits)	Data, set by ENET*n*_MMFR[DATA]. Written to or read from the PHY

43.6.18.2 MDIO clause 45 frame format

The extended MDIO frame structure defined in IEEE 802.3 Clause 45 introduces indirect addressing. First, a write transaction to an address register is done, followed by a write or read transaction which will put the 16-bit data in the register or retrieve the register contents respectively. A preamble of 32 bits of logical ones is sent prior to every transaction. The MDIO data signal is tri-stated between frames.

The extended MDIO defines four transactions, which are determined by the two-bit opcode field.

Table 43-43. MDIO clause 45 frame structure

ST	OP	PRTAD	DEVAD	TA	ADDR/DATA

All bits are transmitted from left to right (Preamble bits first) and all fields have their Most-Significant bit sent first (leftmost in above table). The complete frame has a length of 64 bits (32-bit preamble, 14-bit command, 2-bit bus direction change, 16-bit data). Each bit is transferred with the rising edge of the MDIO clock (MDC).

The fields and transactions are summarized in the following tables.

Table 43-44. MDIO clause 45 frame field descriptions

Field	Description
ST	Start indication. Indicates the end of the preamble and start of the frame. This value is 00 for extended MDIO (Clause 45) frames.
OP	Opcode defines if a read or write operation is performed and is programmed with ENET*n*_MMFR[OP]. See Table 43-45 for more information. 00 Address write 01 Write operation 10 Read inc. operation 11 Read operation
PRTAD	The port address specifies a MDIO port. Each Port can have up to 32 devices which each can have a separate set of registers.
DEVAD	Device address. Up to 32 devices can be addressed (within a port).
TA	Turnaround time, programmed with ENET*n*_MMFR[TA]. Two bit-times are reserved for read operations to switch the data bus from write to read. The PHY device presents its register contents in the data phase and drives the bus from the second bit of the turnaround phase.
ADDR/DATA	16-bit address (for address write) or data, set by ENET*n*_MMFR[DATA], written to or read from the PHY.

Table 43-45. MDIO Clause 45 Transactions

Transaction Type	Description
Address	A write transaction to the internal address register of the device/port. The data section of the frame contains the value to be stored in the device's internal address "pointer" register for further transactions.
Write	Data write to a register. The 16 bit data will be written to the register identified by the device-internal address.
Read	Data is read from the register identified by the device-internal address.
Read inc.	Read with address postincrement. The register identified by the device-internal address is read. After this, the device-internal address is incremented. If the address register is all '1' (0xFFFF) no increment is done (i.e. increment does not wrap around).

43.6.18.3 MDIO clock generation

The MDC clock is generated from the internal bus clock (i.e., IPS bus clock) divided by the value programmed in ENET*n*_MSCR[MII_SPEED].

43.6.18.4 MDIO operation

To perform an MDIO access, set the MDIO command register (ENET*n*_MMFR) according to the description provided in MII Management Frame Register (ENETn_MMFR).

To check when the programmed access completes, read the ENET*n*_EIR[MII] field.

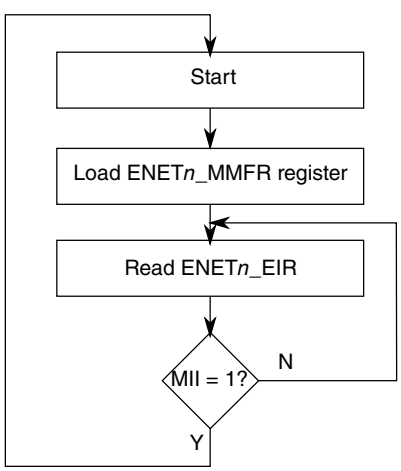


Figure 43-21. MDIO access overview

43.6.19 Ethernet interfaces

The following Ethernet interfaces are implemented:

- Fast Ethernet MII (Media Independent Interface)
- RMII 10/100 using interface converters/gaskets

The following table shows how to configure ENET registers to select each interface.

Mode	RCR[RMII_10T]	RCR[RMII_MODE]
MII - 10 Mbit/s	—	0
MII - 100 Mbit/s	—	0
RMII - 10 Mbit/s	1	1
RMII - 100 Mbit/s	0	1

43.6.19.1 RMII interface

In RMII receive mode, for normal reception following assertion of CRS_DV, RXD[1:0] is 00 until the receiver determines that the receive event has a proper start-of-stream delimiter (SSD).

The preamble appears (RXD[1:0]=01) and the MACs begin capturing data following detection of SFD.

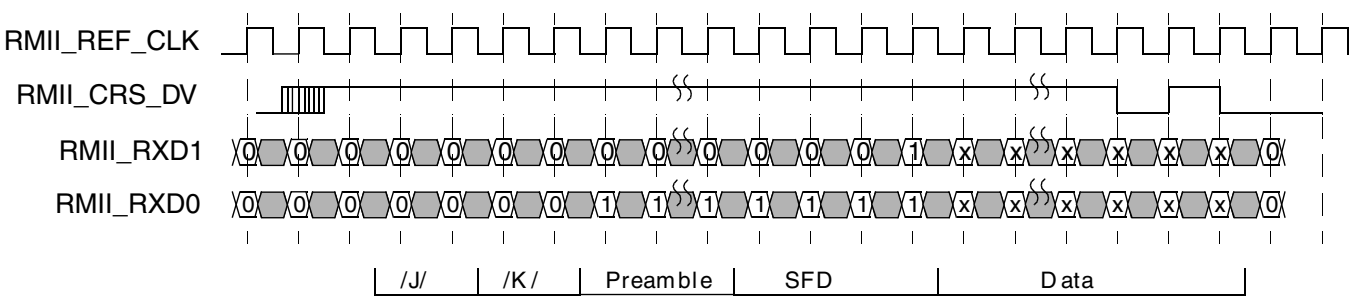


Figure 43-22. RMII receive operation

If a false carrier is detected (bad SSD), then RXD[1:0] is 10 until the end of the receive event. This is a unique pattern since a false carrier can only occur at the beginning of a packet where the preamble is decoded (RXD[1:0] = 01).

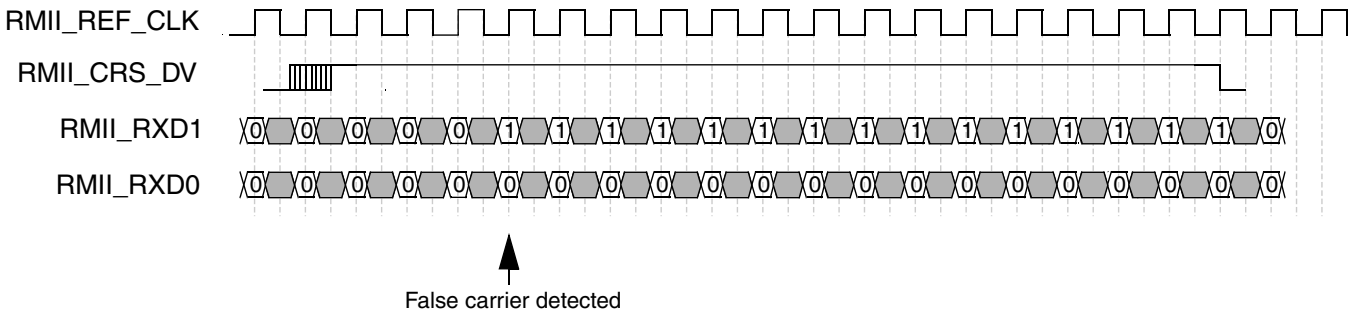


Figure 43-23. RMII receive operation with false carrier

In RMII transmit mode, TXD[1:0] provides valid data for each REF_CLK period while TXEN is asserted.

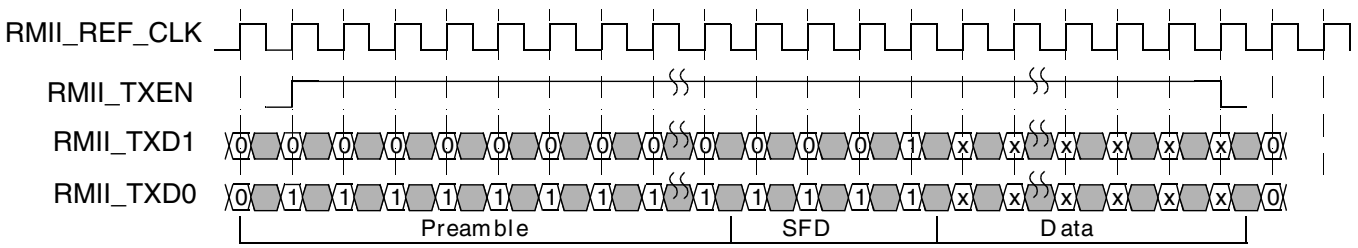


Figure 43-24. RMII transmit operation

43.6.19.2 MII Interface — transmit

On transmit, all data transfers are synchronous to MII_TXCLK rising edge. The MII data enable signal MII_TXEN is asserted to indicate the start of a new frame, and remains asserted until the last byte of the frame is present on the MII_TXD[3:0] bus.

Between frames, MII_TXEN remains deasserted.

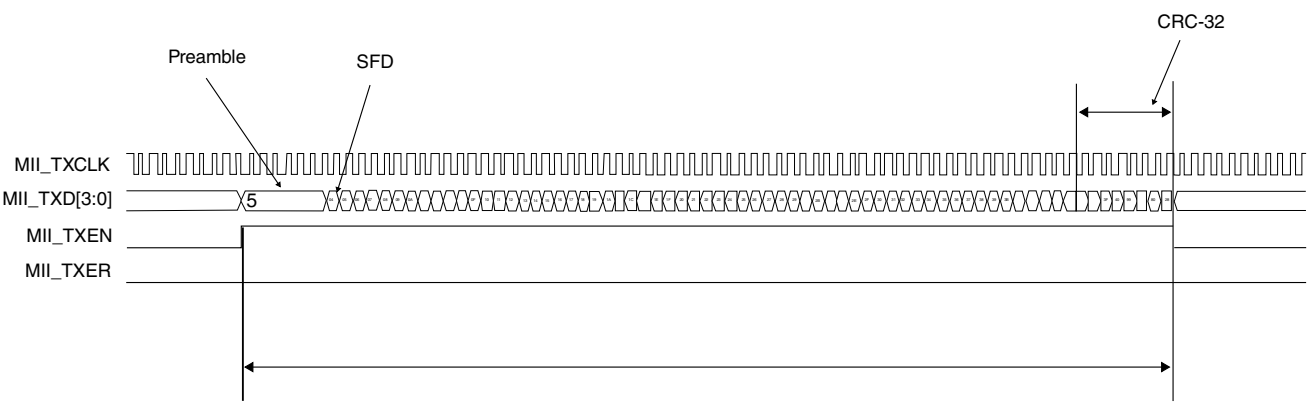


Figure 43-25. MII transmit operation

If a frame is received on the FIFO interface with an error (for example, RxBD[ME] set) the frame is subsequently transmitted with the MII_TXER error signal for one clock cycle at any time during the packet transfer.

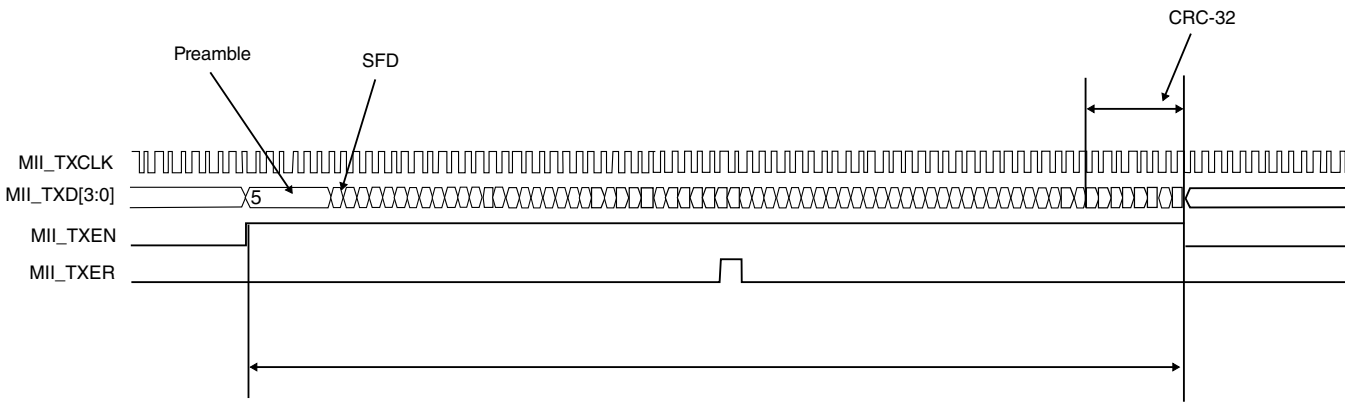


Figure 43-26. MII transmit operation — errored frame

43.6.19.2.1 Transmit with collision — half-duplex

When a collision is detected during a frame transmission (MII_COL asserted), the MAC stops the current transmission, sends a 32-bit jam pattern, and re-transmits the current frame.

(See Collision detection in half-duplex mode for details)

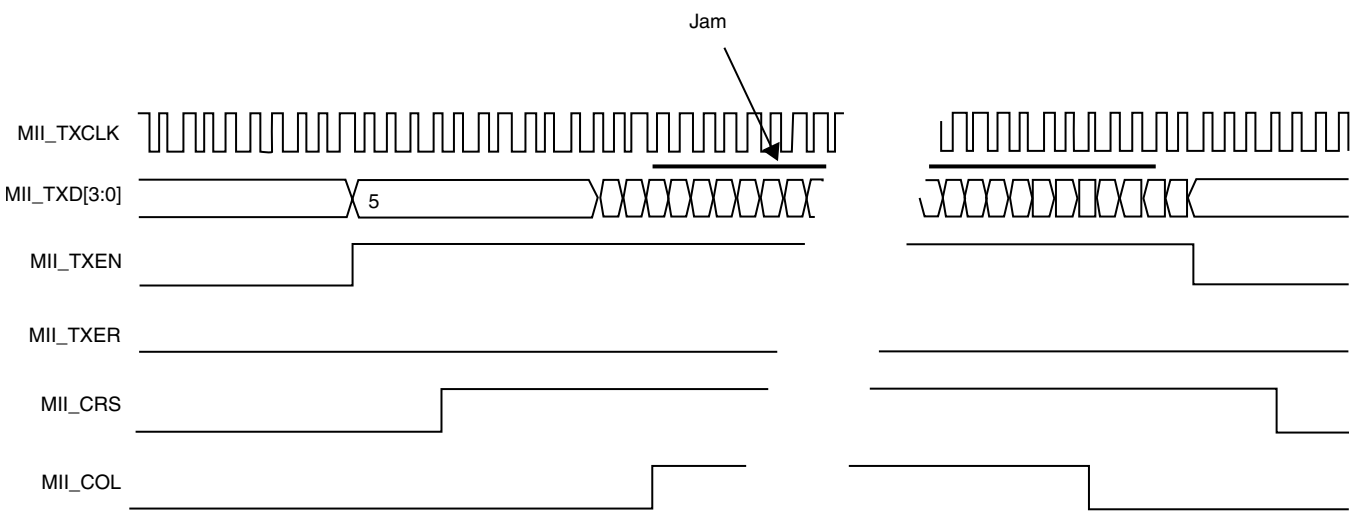


Figure 43-27. MII transmit operation — transmission with collision

43.6.19.3 MII interface — receive

On receive, all signals are sampled on the MII_RXCLK rising edge. The MII data enable signal, MII_RXDV, is asserted by the PHY to indicate the start of a new frame and remains asserted until the last byte of the frame is present on MII_RXD[3:0] bus.

Between frames, MII_RXDV remains deasserted.

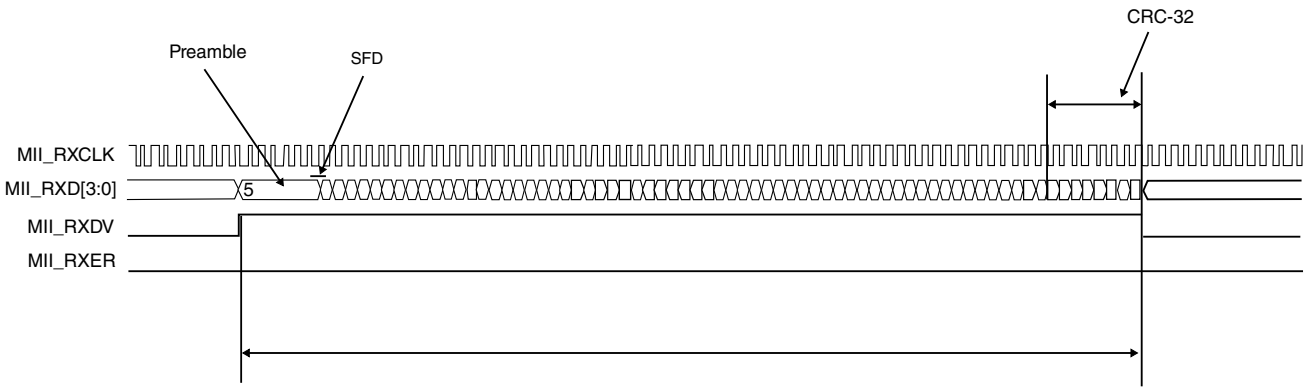


Figure 43-28. MII receive operation

If the PHY detects an error on the frame received from the line, the PHY asserts the MII error signal, MII_RXER, for at least one clock cycle at any time during the packet transfer.

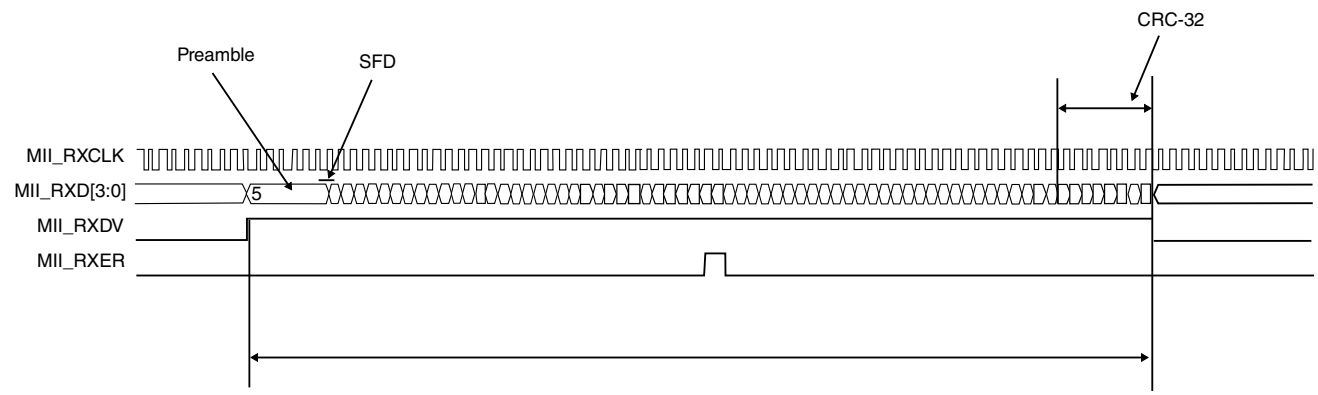


Figure 43-29. MII receive operation — errored frame

A frame received on the MII interface with a PHY error indication is subsequently transferred on the FIFO interface with RxBD[ME] set.

43.6.20 AVB configuration

The following steps give an example of how to initialize the ENET module for AVB.

1. Set up ENET_QOS:
   - Set TX_SCHEME to 000b, credit-based scheme.
   - Set RX_FLUSH0 to 1, enable RX flush for ring 0.
2. Set up TX BD ring and RX BD ring for each queue, 0-2.
   - Program ENET_MRBR, ENET_TDSR, and ENET_RDSR.
   - Program ENET_MRBR1, ENET_TDSR1, and ENET_RDSR1.
   - Program ENET_MRBR2, ENET_TDSR2, and ENET_RDSR2.
   - Program each TX and RX BD ring queue for all classes used in memory.

NOTE

If using credit-based scheme, ensure that enhanced transmit buffer descriptor FTYPE field matches BD ring queue, for example:
- FTYPE = 0h corresponds to ENET_TDSR
- FTYPE = 1h corresponds to ENET_TDSR1
- FTYPE = 2h corresponds to ENET_TDSR2

3. Program ENET_DMA1CFG and ENET_DMA2CFG for class 1 and class 2 corresponding to BD ring queue 1 and 2. DMA_CLASS_EN must be set to one for that class to be enabled. See DMA Class Based Configuration (ENET_DMA*n*CFG) for information on how to program IDLE_SLOPE.
4. Program ENET_RCMR1 and ENET_RCMR2 for class 1 and class 2 matching for receive.

NOTE

Even if a match occurs, if ENET_DMA*n*CFG[DMA_CLASS_EN] is zero for the corresponding class, the RX frame will be automatically forwarded to BD ring queue 0.

5. Program the other ENET registers according to application requirements.
6. Program ENET_RDAR, ENET_RDAR1, ENET_RDAR2, ENET_TDAR, ENET_TDAR1, and ENET_TDAR2 to 0100_0000h according to the classes used.
7. Set ENET_ECR[ETHEREN] to one.

43.6.21 Interrupt coalescence

The purpose of the interrupt coalescing is to reduce the number of interrupts generated by the MAC so as to reduce the CPU loading.

To facilitate this interrupt coalescing for each queue, these registers are available with the same control and configuration fields.

- Transmit Interrupt Coalescing Register (ENET_TXIC*n*) where n=0,1,2 for queue/class 0,1,2.
- Receive Interrupt Coalescing Register (ENET_RXIC*n*) where n=0,1,2 for queue/class 0,1,2.

When coalescing is enabled by asserting the corresponding ICEN field and such interrupt is also enabled by the corresponding interrupt mask of the EIMR register, the MAC generates an interrupt when the threshold number of frames is reached (defined by ICFT) or when the threshold timer expires (defined by ICTT).

When coalescing is disabled by de-asserting ICEN, but interrupt is enabled by the corresponding interrupt mask of the EIMR register, the MAC generates an interrupt as they are received without using coalescing. Interrupt coalescing is done for each transmit and receive queue/class independently.

43.6.21.1 Interrupt coalescence setup

Interrupt coalescence supports both legacy and enhanced BDs. The following guidelines are recommended when setting up interrupt coalescence.

- When the MAC is configured for enhanced (IEEE 1588) mode, that is, enhanced BDs:
  - Set the INT bit in the enhanced received buffer descriptor to one.
  - Set the INT bit in the enhanced transmit buffer descriptor(s) to one.
- Clear the RXB, RXB1, and RXB2 fields in the EIMR register.
- Clear the TXB, TXB1, and TXB2 fields in the EIMR register.

43.6.21.2 Updating the frame count threshold on-the-fly

To update the ICFT field in the RXIC*n* and TXIC*n* registers:

1. Disable interrupt coalescence by clearing the appropriate ICEN field. This will allow the internal interrupt coalescence counter to reset to zero.

NOTE

When disabling interrupt coalescence, if an interrupt event is pending, that is, the interrupt counter is not zero, then an interrupt will occur.

2. Write the new threshold value to the ICFT field.
3. Set ICEN to one.

NOTE

The ICFT field can be updated on-the-fly without disabling the ICEN field. The hardware interrupt will continue and there is a

possibility that an interrupt will occur depending on the state of the hardware counter and the previous ICFT value.

43.6.21.3 Updating the timer threshold on-the-fly

To update the ICTT field in the RXIC*n* and TXIC*n* registers:

1. Disable interrupt coalescence by clearing the appropriate ICEN field. This will allow the internal interrupt coalescence counter to reset to zero.

NOTE

When disabling interrupt coalescence, if an interrupt event is pending, that is, the interrupt counter is not zero, then an interrupt will occur.

2. Write the new timer value to the ICTT field.
3. Set ICEN to one.

Chapter 44
FlexCAN

44.1 Chip-specific FlexCAN information

44.1.1 FlexCAN configuration

This device contains 8 Controller Area Network (CAN) modules: FlexCAN[0:7].

All 8 instances of CAN support FD feature.

FlexCAN_0 includes a new FlexCAN module feature called Pretended Networking mode that addresses the low power requirements of applications. For a CAN node, this mode enables a local approach whereby it can decide when and how it switches into a power saving mode, reducing power consumption and wake-up response time. The Pretended Networking mode is in contrast to partial networking, which would control node clusters within an application.

FlexCAN_0 supports additional filtering (including range) for Pretended Networking. FlexCAN_0 supports Pretended Networking functionality in STOP and LPU_STOP modes.

Each FlexCAN module supports DMA requests from the RxFIFO. Each FlexCAN instantiation supports 96 Message Buffers.

Each FlexCAN module has one external output CAN_Tx (CAN Transmit) and one external input CAN_Rx (CAN Receive).

FLEXCAN_0 supports continuous operation across LPU_RUN mode with FLEXCAN0 CHI running on FXOSC selected via MC_CGM_AC9_SC[SELCTL]. For this, LPU_RUN_CF[CAN0_CONT] needs to be programmed. If CAN0 is operating with this configuration and FXOSC is disabled in between then the Software would not be able to read the CAN0 registers. Hence FXOSC should be kept enabled throughout the CAN0 operation in this configuration.

44.1.1.1 FlexCAN_0 Registers

The Pretended Network Registers are available for FlexCAN_0 instances only.

44.1.1.2 FlexCAN_1 to FlexCAN_7 Registers

The following table shows the address offsets and implementation of the FlexCAN_1 to FlexCAN_7 registers in this chip.

Table 44-1. Register access and reset information

Register	Access type	Affected by hard reset	Affected by soft reset
Module Configuration Register (MCR)	S	Yes	Yes
Control 1 register (CTRL1)	S/U	Yes	No
Free Running Timer register (TIMER)	S/U	Yes	Yes
Rx Mailboxes Global Mask register (RXMGMASK)	S/U	No	No
Rx Buffer 14 Mask register (RX14MASK)	S/U	No	No
Rx Buffer 15 Mask register (RX15MASK)	S/U	No	No
Error Counter Register (ECR)	S/U	Yes	Yes
Error and Status 1 Register (ESR1)	S/U	Yes	Yes
Interrupt Masks 1 register (IMASK1)	S/U	Yes	Yes
Interrupt Flags 1 register (IFLAG1)	S/U	Yes	Yes
Control 2 Register (CTRL2)	S/U	Yes	No
Error and Status 2 Register (ESR2)	S/U	Yes	Yes
CRC Register (CRCR)	S/U	Yes	Yes
Rx FIFO Global Mask register (RXFGMASK)	S/U	No	No
Rx FIFO Information Register (RXFIR)	S/U	No	No
Message buffers	S/U	No	No
Rx Individual Mask Registers	S/U	No	No

44.1.2 Requirements for entering FlexCAN modes: Freeze, Module Disable, Stop

Follow the procedure described in Freeze mode entry to place the FlexCAN module in that mode.

On this chip, the FlexCAN module cannot directly enter:

- Module Disable mode
- Stop mode

To enter these modes, the FlexCAN module must first enter Freeze mode. See the following information about each of the modes.

44.1.2.1 Freeze mode entry

See Freeze mode for details about this mode.

On this chip, the procedure to enter Freeze mode is:

1. Set both CAN_MCR[FRZ] (Freeze Enable) and CAN_MCR[HALT] (Halt) to 1.
2. Check whether CAN_MCR[MDIS] (Module Disable) is set to 1. If it is, clear it to 0.
3. Poll the MCR register until CAN_MCR[FRZACK] (Freeze Mode Acknowledge) is set to 1 or the timeout is reached.

NOTE

The minimum timeout duration must be equivalent to:

a. 730 CAN Nominal bits if CAN_MCR[FDEN] (CAN FD Operation Enable) is set to 1 (CAN bits calculated at arbitration bit rate),
b. 180 CAN bits if CAN_MCR[FDEN] is cleared to 0.

4. If CAN_MCR[FRZACK] is set to 1, no further action is required. Skip steps 5 to 8.
5. If the timeout is reached because CAN_MCR[FRZACK] remains cleared to 0, then set CAN_MCR[SOFTRST] (Soft Reset) to 1.
6. Poll MCR until CAN_MCR[SOFTRST] is cleared to 0.
7. Reconfigure the Module Control Register (CAN_MCR).
8. Reconfigure all the Interrupt Mask Registers (CAN_IMASK*n*).

After these steps are complete, the module is in Freeze mode.

44.1.2.2 Module Disable mode entry

See Module Disable mode for details about this mode.

CAUTION

In the detailed description of this mode, the list of FlexCAN behavior that begins with "If the module is disabled during transmission or reception" does not apply to this chip.

On this chip, the procedure to enter Module Disable mode is:

1. Enter Freeze mode by following the procedure described in Freeze mode entry.
2. Request Module Disable mode by setting CAN_MCR[MDIS] to 1.
3. Poll CAN_MCR until CAN_MCR[LPMACK] is set to 1. The minimum timeout duration for setting CAN_MCR[LPMACK] is 2 CAN Nominal bits.

44.1.2.3 Stop mode entry

See Stop mode for details about this mode.

CAUTION

In the detailed description of this mode, the list of FlexCAN behavior that begins with "If Stop mode is requested during transmission or reception" does not apply to this chip.

On this chip, the procedure to enter Stop mode is:

1. Enter Freeze mode by following the procedure described in Freeze mode entry.
2. Request Stop mode.
3. Poll MCR until CAN_MCR[LPMACK] is set to 1. The minimum timeout duration for setting CAN_MCR[LPMACK] is 2 CAN Nominal bits.

44.1.3 External time tick

The external time tick for the Free Running Timer counter is given by PIT-RTI as per the following table for the multiple instances of FlexCAN.

Table 44-2. External time tick

FlexCAN Instance	PIT Channel
FlexCAN_0, FlexCAN_1, FlexCAN_2, FlexCAN_3	PIT_14
FlexCAN_4, FlexCAN_5, FlexCAN_6, FlexCAN_7	PIT_15

44.1.4 FlexCAN's stop-stop ack protocol

- If FlexCAN is disabled, it acknowledges the stop request immediately.
- If FlexCAN is enabled, it will acknowledge the stop request after all current Tx or Rx process is finished.

Acknowledgement for stop mode request will be asserted within 180 CAN bits from the low-power mode request by the core, and negated within 2 CAN bits after the low-power mode request removal (see Protocol timing). When FlexCAN is in Pretended Networking mode acknowledgement will be negated within 180 CAN bits after the low-power mode request removal.

44.2 Introduction

The FlexCAN module is a communication controller implementing the CAN protocol according to the ISO 11898-1 standard and CAN 2.0 B protocol specifications. A general block diagram is shown in the following figure, which describes the main subblocks implemented in the FlexCAN module, including one associated memory for storing message buffers, Receive Global Mask registers, Receive Individual Mask registers, Receive FIFO filters, and Receive FIFO ID filters. The functions of the submodules are described in subsequent sections.

NOTE

Rx FIFOs cannot be used in FD mode.

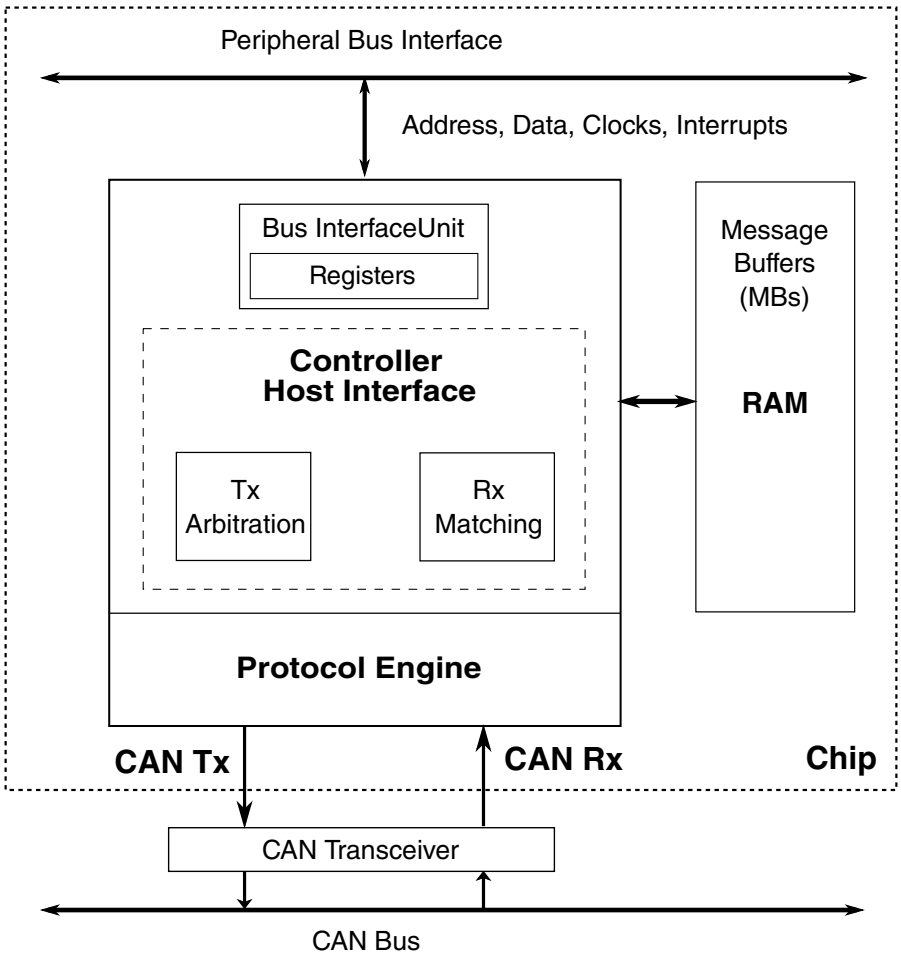


Figure 44-1. FlexCAN block diagram

44.2.1 Overview

The CAN protocol was primarily designed to be used as a vehicle serial data bus, meeting the specific requirements of this field:

- Real-time processing
- Reliable operation in the EMI environment of a vehicle
- Cost-effectiveness
- Required bandwidth

The FlexCAN module is a full implementation of the CAN protocol specification, the CAN with Flexible Data rate (CAN FD) protocol and the CAN 2.0 version B protocol, which supports both standard and extended message frames and long payloads up to 64 bytes transferred at faster rates up to 8 Mbps. The message buffers are stored in an embedded RAM dedicated to the FlexCAN module. See the chip configuration details for the actual number of message buffers configured in the chip.

The Protocol Engine (PE) submodule manages the serial communication on the CAN bus:

- Requesting RAM access for receiving and transmitting message frames
- Validating received messages
- Performing error handling
- Detecting CAN FD messages

The Controller Host Interface (CHI) sub-module handles message buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms for both CAN FD and non-CAN FD message formats.

The Bus Interface Unit (BIU) sub-module controls the access to and from the internal interface bus, in order to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs, DMA and test signals are accessed through the BIU.

44.2.2 FlexCAN module features

The FlexCAN module includes these distinctive features:

- Full implementation of the CAN with Flexible Data Rate (CAN FD) protocol specification and CAN protocol specification, Version 2.0 B
    - Standard data frames
    - Extended data frames
    - Zero to sixty four bytes data length
    - Programmable bit rate (see the chip-specific FlexCAN information for the specific maximum rate configuration)
    - Content-related addressing
- Compliant with the ISO 11898-1 standard
- Flexible mailboxes configurable to store 0 to 8, 16, 32 or 64 bytes data length
- Each mailbox configurable as receive or transmit, all supporting standard and extended messages
- Individual Rx Mask registers per mailbox
- Full-featured Rx FIFO with storage capacity for up to six frames and automatic internal pointer handling with DMA support

- Transmission abort capability
- Flexible message buffers (MBs), totaling 96 message buffers of 8 bytes data length each, configurable as Rx or Tx
- Programmable clock source to the CAN Protocol Interface, either peripheral clock or oscillator clock
- RAM not used by reception or transmission structures can be used as general purpose RAM space
- Listen-Only mode capability
- Programmable Loop-Back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free-running timer, with an optional external time tick
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independence from the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes, with programmable wake up on bus activity or matching with received frames (Pretended Networking)
- Transceiver Delay Compensation feature when transmitting CAN FD messages at faster data rates
- Remote request frames may be handled automatically or by software
- CAN bit time settings and configuration bits can only be written in Freeze mode
- Tx mailbox status (Lowest priority buffer or empty buffer)
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- SYNCH bit available in Error in Status 1 register to inform that the module is synchronous with CAN bus
- CRC status for transmitted message
- Rx FIFO Global Mask register
- Selectable priority between mailboxes and Rx FIFO during matching process

- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard, or 512 partial (8 bit) IDs, with up to 32 individual masking capability
- 100% backward compatibility with previous FlexCAN version
- Supports Pretended Networking functionality in low power: Stop mode

44.2.3 Modes of operation

The FlexCAN module has these functional modes:

- Normal mode (User or Supervisor):

  In Normal mode, the module operates receiving and/or transmitting message frames, errors are handled normally, and all CAN Protocol functions are enabled. User and Supervisor Modes differ in the access to some restricted control registers.

- Freeze mode:

  Freeze mode is enabled when the FRZ bit in MCR is asserted. If enabled, Freeze mode is entered when MCR[HALT] is set or when Debug mode is requested at chip level and MCR[FRZ_ACK] is asserted by the FlexCAN. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Freeze mode for more information.

- Loop-Back mode:

  The module enters this mode when the LPB field in the Control 1 Register is asserted. In this mode, FlexCAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic '1'). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

- Listen-Only mode:

  The module enters this mode when the LOM field in the Control 1 Register is asserted. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN Error Passive mode. Only messages acknowledged by

another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- CAN FD Active mode:

  In this mode, FlexCAN is capable of transmitting and receiving all messages formatted according to the CAN FD Protocol and CAN 2.0 Protocol 2.0 in a interleaved fashion. The CPU can set the FlexCAN into CAN FD Active mode by configuring the MCR[FDEN] bit field in Freeze Mode.

For low-power operation, the FlexCAN module has:

- Module Disable mode:

  This low-power mode is entered when the MDIS bit in the MCR Register is asserted by the CPU and the LPM_ACK is asserted by the FlexCAN. When disabled, the module requests to disable the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Exit from this mode is done by negating the MDIS bit in the MCR register. See Module Disable mode for more information.

- Stop mode:

  This low power mode is entered when Stop mode is requested at chip level and the LPM_ACK bit in the MCR Register is asserted by the FlexCAN. When in Stop Mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. Exit from this mode happens when the Stop mode request is removed, or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled. See Stop mode for more information.

- Pretended Network Mode:

  This mode can be selected to operate together with Stop mode. Before entering in one of these low power modes the PNET_EN bit in MCR Register must be asserted. Once in low power mode, CHI sub-block clocks are shut down and CAN_PE sub-block is kept clocked, so that the Rx receive process is still active to filter incoming messages (see Receive Process under Pretended Networking Mode) as defined by the configuration registers (see Pretended Networking Control 1 Register (CTRL1_PN)). Upon detecting a wake up event, a Wake Up interrupt is issued to the system. When PNET_EN bit in MCR is asserted, the CPU must disable the Self Wake Up feature by negating SLF_WAK bit in MCR register (see Module Configuration Register (MCR)).

44.3 FlexCAN signal descriptions

The FlexCAN module has two I/O signals connected to the external chip pins. These signals are summarized in the following table and described in more detail in the next subsections.

Table 44-3. FlexCAN signal descriptions

Signal	Description	I/O
CAN Rx	CAN Receive Pin	Input
CAN Tx	CAN Transmit Pin	Output

44.3.1 CAN Rx

This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

44.3.2 CAN Tx

This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

44.4 Memory map/register definition

This section describes the registers and data structures in the FlexCAN module. The base address of the module depends on the particular memory map of the chip.

44.4.1 FlexCAN memory mapping

The memory map for the FlexCAN module is shown in the following table.

The address space occupied by FlexCAN has 128 bytes for registers starting at the module base address, followed by embedded RAM starting at address offset 0x0080.

Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV field in the MCR register. These registers are identified as S/U in the Access column of Table 44-4.

Table 44-4. Register access and reset information

Register	Access type	Affected by hard reset	Affected by soft reset
Module Configuration Register (CAN_MCR)	S	Yes	Yes
Control 1 register (CAN_CTRL1)	S/U	Yes	No
Free Running Timer register (CAN_TIMER)	S/U	Yes	Yes
Rx Mailboxes Global Mask register (CAN_RXMGMASK)	S/U	No	No
Rx Buffer 14 Mask register (CAN_RX14MASK)	S/U	No	No
Rx Buffer 15 Mask register (CAN_RX15MASK)	S/U	No	No
Error Counter Register (CAN_ECR)	S/U	Yes	Yes
Error and Status 1 Register (CAN_ESR1)	S/U	Yes	Yes
Interrupt Masks 2 register (CAN_IMASK2)	S/U	Yes	Yes
Interrupt Masks 1 register (CAN_IMASK1)	S/U	Yes	Yes
Interrupt Flags 2 register (CAN_IFLAG2)	S/U	Yes	Yes
Interrupt Flags 1 register (CAN_IFLAG1)	S/U	Yes	Yes
Control 2 Register (CAN_CTRL2)	S/U	Yes	No
Error and Status 2 Register (CAN_ESR2)	S/U	Yes	Yes
CRC Register (CAN_CRCR)	S/U	Yes	Yes
Rx FIFO Global Mask register (CAN_RXFGMASK)	S/U	No	No
Rx FIFO Information Register (CAN_RXFIR)	S/U	No	No
CAN Bit Timing Register (CAN_CBT)	S/U	Yes	No
Interrupt Masks 3 register (CAN_IMASK3)	S/U	Yes	Yes
Interrupt Flags 3 register (CAN_IFLAG3)	S/U	Yes	Yes
Message buffers	S/U	No	No
Rx Individual Mask Registers	S/U	No	No
Pretended Networking Control 1 register (CAN_CTRL1_PN)	S/U	Yes	Yes
Pretended Networking Control 2 register (CAN_CTRL2_PN)	S/U	Yes	Yes
Pretended Networking Wake Up Match register (CAN_WU_MTC)	S/U	Yes	Yes
Pretended Networking ID Filter 1 Register (CAN_FLT_ID1)	S/U	Yes	Yes
Pretended Networking DLC Filter register (CAN_FLT_DLC)	S/U	Yes	Yes
Pretended Networking Payload Low Filter 1 register (CAN_PL1_LO)	S/U	Yes	Yes
Pretended Networking Payload High Filter 1 register (CAN_PL1_HI)	S/U	Yes	Yes
Pretended Networking ID Filter 2 Register / ID Mask register (CAN_FLT_ID2_IDMASK)	S/U	Yes	Yes
Pretended Networking Payload Low Filter 2 Register / Payload Low Mask Register (CAN_PL2_PLMASK_LO)	S/U	Yes	Yes

Table continues on the next page...

Table 44-4. Register access and reset information (continued)

Register	Access type	Affected by hard reset	Affected by soft reset
Pretended Networking Payload High Filter 2 Register / Payload High Mask Register (CAN_PL2_PLMASK_HI)	S/U	Yes	Yes
Pretended Networking Wake Up Message Buffer 0 register (CAN_WMB0)	S/U	Yes	No
Pretended Networking Wake Up Message Buffer 1 register (CAN_WMB1)	S/U	Yes	No
Pretended Networking Wake Up Message Buffer 2 register (CAN_WMB2)	S/U	Yes	No
Pretended Networking Wake Up Message Buffer 3 register (CAN_WMB3)	S/U	Yes	No
CAN FD Control register (CAN_FDCTRL)	S/U	Yes	No
CAN FD Bit Timing register (CAN_FDCBT)	S/U	Yes	No
CAN FD CRC register (CAN_FDCRC)	S/U	Yes	Yes

The FlexCAN module can store CAN messages for transmission and reception using mailboxes and Rx FIFO structures.

The table below shows the FlexCAN memory map.

The address range from offset 0x80 to 0x67F allocates the ninety-six 128-bit Message Buffers (MBs).

The memory maps for the message buffers are in FlexCAN message buffer memory map.

CAN memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (CAN_MCR)	32	R/W	See section	44.4.2/1542
4	Control 1 register (CAN_CTRL1)	32	R/W	0000_0000h	44.4.3/1547
8	Free Running Timer (CAN_TIMER)	32	R/W	0000_0000h	44.4.4/1551
10	Rx Mailboxes Global Mask Register (CAN_RXMGMASK)	32	R/W	Undefined	44.4.5/1552
14	Rx 14 Mask register (CAN_RX14MASK)	32	R/W	Undefined	44.4.6/1553
18	Rx 15 Mask register (CAN_RX15MASK)	32	R/W	Undefined	44.4.7/1554
1C	Error Counter (CAN_ECR)	32	R/W	0000_0000h	44.4.8/1555
20	Error and Status 1 register (CAN_ESR1)	32	R/W	See section	44.4.9/1557
24	Interrupt Masks 2 register (CAN_IMASK2)	32	R/W	0000_0000h	44.4.10/ 1564
28	Interrupt Masks 1 register (CAN_IMASK1)	32	R/W	0000_0000h	44.4.11/ 1564
2C	Interrupt Flags 2 register (CAN_IFLAG2)	32	R/W	0000_0000h	44.4.12/ 1565
30	Interrupt Flags 1 register (CAN_IFLAG1)	32	R/W	0000_0000h	44.4.13/ 1566

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	Control 2 register (CAN_CTRL2)	32	R/W	See section	44.4.14/ 1569
38	Error and Status 2 register (CAN_ESR2)	32	R/W	0000_0000h	44.4.15/ 1572
44	CRC Register (CAN_CRCR)	32	R	0000_0000h	44.4.16/ 1574
48	Rx FIFO Global Mask register (CAN_RXFGMASK)	32	R/W	Undefined	44.4.17/ 1574
4C	Rx FIFO Information Register (CAN_RXFIR)	32	R	Undefined	44.4.18/ 1575
50	CAN Bit Timing Register (CAN_CBT)	32	R/W	See section	44.4.19/ 1576
6C	Interrupt Masks 3 Register (CAN_IMASK3)	32	R/W	0000_0000h	44.4.20/ 1578
74	Interrupt Flags 3 Register (CAN_IFLAG3)	32	R/W	0000_0000h	44.4.21/ 1579
880	Rx Individual Mask Registers (CAN_RXIMR0)	32	R/W	Undefined	44.4.22/ 1579
884	Rx Individual Mask Registers (CAN_RXIMR1)	32	R/W	Undefined	44.4.22/ 1579
888	Rx Individual Mask Registers (CAN_RXIMR2)	32	R/W	Undefined	44.4.22/ 1579
88C	Rx Individual Mask Registers (CAN_RXIMR3)	32	R/W	Undefined	44.4.22/ 1579
890	Rx Individual Mask Registers (CAN_RXIMR4)	32	R/W	Undefined	44.4.22/ 1579
894	Rx Individual Mask Registers (CAN_RXIMR5)	32	R/W	Undefined	44.4.22/ 1579
898	Rx Individual Mask Registers (CAN_RXIMR6)	32	R/W	Undefined	44.4.22/ 1579
89C	Rx Individual Mask Registers (CAN_RXIMR7)	32	R/W	Undefined	44.4.22/ 1579
8A0	Rx Individual Mask Registers (CAN_RXIMR8)	32	R/W	Undefined	44.4.22/ 1579
8A4	Rx Individual Mask Registers (CAN_RXIMR9)	32	R/W	Undefined	44.4.22/ 1579
8A8	Rx Individual Mask Registers (CAN_RXIMR10)	32	R/W	Undefined	44.4.22/ 1579
8AC	Rx Individual Mask Registers (CAN_RXIMR11)	32	R/W	Undefined	44.4.22/ 1579
8B0	Rx Individual Mask Registers (CAN_RXIMR12)	32	R/W	Undefined	44.4.22/ 1579
8B4	Rx Individual Mask Registers (CAN_RXIMR13)	32	R/W	Undefined	44.4.22/ 1579

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8B8	Rx Individual Mask Registers (CAN_RXIMR14)	32	R/W	Undefined	44.4.22/ 1579
8BC	Rx Individual Mask Registers (CAN_RXIMR15)	32	R/W	Undefined	44.4.22/ 1579
8C0	Rx Individual Mask Registers (CAN_RXIMR16)	32	R/W	Undefined	44.4.22/ 1579
8C4	Rx Individual Mask Registers (CAN_RXIMR17)	32	R/W	Undefined	44.4.22/ 1579
8C8	Rx Individual Mask Registers (CAN_RXIMR18)	32	R/W	Undefined	44.4.22/ 1579
8CC	Rx Individual Mask Registers (CAN_RXIMR19)	32	R/W	Undefined	44.4.22/ 1579
8D0	Rx Individual Mask Registers (CAN_RXIMR20)	32	R/W	Undefined	44.4.22/ 1579
8D4	Rx Individual Mask Registers (CAN_RXIMR21)	32	R/W	Undefined	44.4.22/ 1579
8D8	Rx Individual Mask Registers (CAN_RXIMR22)	32	R/W	Undefined	44.4.22/ 1579
8DC	Rx Individual Mask Registers (CAN_RXIMR23)	32	R/W	Undefined	44.4.22/ 1579
8E0	Rx Individual Mask Registers (CAN_RXIMR24)	32	R/W	Undefined	44.4.22/ 1579
8E4	Rx Individual Mask Registers (CAN_RXIMR25)	32	R/W	Undefined	44.4.22/ 1579
8E8	Rx Individual Mask Registers (CAN_RXIMR26)	32	R/W	Undefined	44.4.22/ 1579
8EC	Rx Individual Mask Registers (CAN_RXIMR27)	32	R/W	Undefined	44.4.22/ 1579
8F0	Rx Individual Mask Registers (CAN_RXIMR28)	32	R/W	Undefined	44.4.22/ 1579
8F4	Rx Individual Mask Registers (CAN_RXIMR29)	32	R/W	Undefined	44.4.22/ 1579
8F8	Rx Individual Mask Registers (CAN_RXIMR30)	32	R/W	Undefined	44.4.22/ 1579
8FC	Rx Individual Mask Registers (CAN_RXIMR31)	32	R/W	Undefined	44.4.22/ 1579
900	Rx Individual Mask Registers (CAN_RXIMR32)	32	R/W	Undefined	44.4.22/ 1579
904	Rx Individual Mask Registers (CAN_RXIMR33)	32	R/W	Undefined	44.4.22/ 1579
908	Rx Individual Mask Registers (CAN_RXIMR34)	32	R/W	Undefined	44.4.22/ 1579
90C	Rx Individual Mask Registers (CAN_RXIMR35)	32	R/W	Undefined	44.4.22/ 1579

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
910	Rx Individual Mask Registers (CAN_RXIMR36)	32	R/W	Undefined	44.4.22/ 1579
914	Rx Individual Mask Registers (CAN_RXIMR37)	32	R/W	Undefined	44.4.22/ 1579
918	Rx Individual Mask Registers (CAN_RXIMR38)	32	R/W	Undefined	44.4.22/ 1579
91C	Rx Individual Mask Registers (CAN_RXIMR39)	32	R/W	Undefined	44.4.22/ 1579
920	Rx Individual Mask Registers (CAN_RXIMR40)	32	R/W	Undefined	44.4.22/ 1579
924	Rx Individual Mask Registers (CAN_RXIMR41)	32	R/W	Undefined	44.4.22/ 1579
928	Rx Individual Mask Registers (CAN_RXIMR42)	32	R/W	Undefined	44.4.22/ 1579
92C	Rx Individual Mask Registers (CAN_RXIMR43)	32	R/W	Undefined	44.4.22/ 1579
930	Rx Individual Mask Registers (CAN_RXIMR44)	32	R/W	Undefined	44.4.22/ 1579
934	Rx Individual Mask Registers (CAN_RXIMR45)	32	R/W	Undefined	44.4.22/ 1579
938	Rx Individual Mask Registers (CAN_RXIMR46)	32	R/W	Undefined	44.4.22/ 1579
93C	Rx Individual Mask Registers (CAN_RXIMR47)	32	R/W	Undefined	44.4.22/ 1579
940	Rx Individual Mask Registers (CAN_RXIMR48)	32	R/W	Undefined	44.4.22/ 1579
944	Rx Individual Mask Registers (CAN_RXIMR49)	32	R/W	Undefined	44.4.22/ 1579
948	Rx Individual Mask Registers (CAN_RXIMR50)	32	R/W	Undefined	44.4.22/ 1579
94C	Rx Individual Mask Registers (CAN_RXIMR51)	32	R/W	Undefined	44.4.22/ 1579
950	Rx Individual Mask Registers (CAN_RXIMR52)	32	R/W	Undefined	44.4.22/ 1579
954	Rx Individual Mask Registers (CAN_RXIMR53)	32	R/W	Undefined	44.4.22/ 1579
958	Rx Individual Mask Registers (CAN_RXIMR54)	32	R/W	Undefined	44.4.22/ 1579
95C	Rx Individual Mask Registers (CAN_RXIMR55)	32	R/W	Undefined	44.4.22/ 1579
960	Rx Individual Mask Registers (CAN_RXIMR56)	32	R/W	Undefined	44.4.22/ 1579
964	Rx Individual Mask Registers (CAN_RXIMR57)	32	R/W	Undefined	44.4.22/ 1579

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
968	Rx Individual Mask Registers (CAN_RXIMR58)	32	R/W	Undefined	44.4.22/ 1579
96C	Rx Individual Mask Registers (CAN_RXIMR59)	32	R/W	Undefined	44.4.22/ 1579
970	Rx Individual Mask Registers (CAN_RXIMR60)	32	R/W	Undefined	44.4.22/ 1579
974	Rx Individual Mask Registers (CAN_RXIMR61)	32	R/W	Undefined	44.4.22/ 1579
978	Rx Individual Mask Registers (CAN_RXIMR62)	32	R/W	Undefined	44.4.22/ 1579
97C	Rx Individual Mask Registers (CAN_RXIMR63)	32	R/W	Undefined	44.4.22/ 1579
980	Rx Individual Mask Registers (CAN_RXIMR64)	32	R/W	Undefined	44.4.22/ 1579
984	Rx Individual Mask Registers (CAN_RXIMR65)	32	R/W	Undefined	44.4.22/ 1579
988	Rx Individual Mask Registers (CAN_RXIMR66)	32	R/W	Undefined	44.4.22/ 1579
98C	Rx Individual Mask Registers (CAN_RXIMR67)	32	R/W	Undefined	44.4.22/ 1579
990	Rx Individual Mask Registers (CAN_RXIMR68)	32	R/W	Undefined	44.4.22/ 1579
994	Rx Individual Mask Registers (CAN_RXIMR69)	32	R/W	Undefined	44.4.22/ 1579
998	Rx Individual Mask Registers (CAN_RXIMR70)	32	R/W	Undefined	44.4.22/ 1579
99C	Rx Individual Mask Registers (CAN_RXIMR71)	32	R/W	Undefined	44.4.22/ 1579
9A0	Rx Individual Mask Registers (CAN_RXIMR72)	32	R/W	Undefined	44.4.22/ 1579
9A4	Rx Individual Mask Registers (CAN_RXIMR73)	32	R/W	Undefined	44.4.22/ 1579
9A8	Rx Individual Mask Registers (CAN_RXIMR74)	32	R/W	Undefined	44.4.22/ 1579
9AC	Rx Individual Mask Registers (CAN_RXIMR75)	32	R/W	Undefined	44.4.22/ 1579
9B0	Rx Individual Mask Registers (CAN_RXIMR76)	32	R/W	Undefined	44.4.22/ 1579
9B4	Rx Individual Mask Registers (CAN_RXIMR77)	32	R/W	Undefined	44.4.22/ 1579
9B8	Rx Individual Mask Registers (CAN_RXIMR78)	32	R/W	Undefined	44.4.22/ 1579
9BC	Rx Individual Mask Registers (CAN_RXIMR79)	32	R/W	Undefined	44.4.22/ 1579

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9C0	Rx Individual Mask Registers (CAN_RXIMR80)	32	R/W	Undefined	44.4.22/ 1579
9C4	Rx Individual Mask Registers (CAN_RXIMR81)	32	R/W	Undefined	44.4.22/ 1579
9C8	Rx Individual Mask Registers (CAN_RXIMR82)	32	R/W	Undefined	44.4.22/ 1579
9CC	Rx Individual Mask Registers (CAN_RXIMR83)	32	R/W	Undefined	44.4.22/ 1579
9D0	Rx Individual Mask Registers (CAN_RXIMR84)	32	R/W	Undefined	44.4.22/ 1579
9D4	Rx Individual Mask Registers (CAN_RXIMR85)	32	R/W	Undefined	44.4.22/ 1579
9D8	Rx Individual Mask Registers (CAN_RXIMR86)	32	R/W	Undefined	44.4.22/ 1579
9DC	Rx Individual Mask Registers (CAN_RXIMR87)	32	R/W	Undefined	44.4.22/ 1579
9E0	Rx Individual Mask Registers (CAN_RXIMR88)	32	R/W	Undefined	44.4.22/ 1579
9E4	Rx Individual Mask Registers (CAN_RXIMR89)	32	R/W	Undefined	44.4.22/ 1579
9E8	Rx Individual Mask Registers (CAN_RXIMR90)	32	R/W	Undefined	44.4.22/ 1579
9EC	Rx Individual Mask Registers (CAN_RXIMR91)	32	R/W	Undefined	44.4.22/ 1579
9F0	Rx Individual Mask Registers (CAN_RXIMR92)	32	R/W	Undefined	44.4.22/ 1579
9F4	Rx Individual Mask Registers (CAN_RXIMR93)	32	R/W	Undefined	44.4.22/ 1579
9F8	Rx Individual Mask Registers (CAN_RXIMR94)	32	R/W	Undefined	44.4.22/ 1579
9FC	Rx Individual Mask Registers (CAN_RXIMR95)	32	R/W	Undefined	44.4.22/ 1579
B00	Pretended Networking Control 1 Register (CAN_CTRL1_PN)	32	R/W	See section	44.4.23/ 1581
B04	Pretended Networking Control 2 Register (CAN_CTRL2_PN)	32	R/W	0000_0000h	44.4.24/ 1582
B08	Pretended Networking Wake Up Match Register (CAN_WU_MTC)	32	R/W	See section	44.4.25/ 1583
B0C	Pretended Networking ID Filter 1 Register (CAN_FLT_ID1)	32	R/W	See section	44.4.26/ 1585
B10	Pretended Networking DLC Filter Register (CAN_FLT_DLC)	32	R/W	0000_0008h	44.4.27/ 1586
B14	Pretended Networking Payload Low Filter 1 Register (CAN_PL1_LO)	32	R/W	0000_0000h	44.4.28/ 1587

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B18	Pretended Networking Payload High Filter 1 Register (CAN_PL1_HI)	32	R/W	0000_0000h	44.4.29/ 1587
B1C	Pretended Networking ID Filter 2 Register / ID Mask Register (CAN_FLT_ID2_IDMASK)	32	R/W	See section	44.4.30/ 1588
B20	Pretended Networking Payload Low Filter 2 Register / Payload Low Mask Register (CAN_PL2_PLMASK_LO)	32	R/W	0000_0000h	44.4.31/ 1589
B24	Pretended Networking Payload High Filter 2 low order bits / Payload High Mask Register (CAN_PL2_PLMASK_HI)	32	R/W	0000_0000h	44.4.32/ 1590
B40	Wake Up Message Buffer Register for C/S (CAN_WMB0_CS)	32	R	See section	44.4.33/ 1590
B44	Wake Up Message Buffer Register for ID (CAN_WMB0_ID)	32	R	0000_0000h	44.4.34/ 1591
B48	Wake Up Message Buffer Register for Data 0-3 (CAN_WMB0_D03)	32	R	0000_0000h	44.4.35/ 1592
B4C	Wake Up Message Buffer Register Data 4-7 (CAN_WMB0_D47)	32	R	0000_0000h	44.4.36/ 1593
B50	Wake Up Message Buffer Register for C/S (CAN_WMB1_CS)	32	R	See section	44.4.33/ 1590
B54	Wake Up Message Buffer Register for ID (CAN_WMB1_ID)	32	R	0000_0000h	44.4.34/ 1591
B58	Wake Up Message Buffer Register for Data 0-3 (CAN_WMB1_D03)	32	R	0000_0000h	44.4.35/ 1592
B5C	Wake Up Message Buffer Register Data 4-7 (CAN_WMB1_D47)	32	R	0000_0000h	44.4.36/ 1593
B60	Wake Up Message Buffer Register for C/S (CAN_WMB2_CS)	32	R	See section	44.4.33/ 1590
B64	Wake Up Message Buffer Register for ID (CAN_WMB2_ID)	32	R	0000_0000h	44.4.34/ 1591
B68	Wake Up Message Buffer Register for Data 0-3 (CAN_WMB2_D03)	32	R	0000_0000h	44.4.35/ 1592
B6C	Wake Up Message Buffer Register Data 4-7 (CAN_WMB2_D47)	32	R	0000_0000h	44.4.36/ 1593
B70	Wake Up Message Buffer Register for C/S (CAN_WMB3_CS)	32	R	See section	44.4.33/ 1590
B74	Wake Up Message Buffer Register for ID (CAN_WMB3_ID)	32	R	0000_0000h	44.4.34/ 1591
B78	Wake Up Message Buffer Register for Data 0-3 (CAN_WMB3_D03)	32	R	0000_0000h	44.4.35/ 1592
B7C	Wake Up Message Buffer Register Data 4-7 (CAN_WMB3_D47)	32	R	0000_0000h	44.4.36/ 1593
C00	CAN FD Control Register (CAN_FDCTRL)	32	R/W	See section	44.4.37/ 1593
C04	CAN FD Bit Timing Register (CAN_FDCBT)	32	R/W	See section	44.4.38/ 1597

Table continues on the next page...

CAN memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C08	CAN FD CRC Register (CAN_FDCRC)	32	R	0000_0000h	44.4.39/ 1599

44.4.2 Module Configuration Register (CAN_MCR)

This register defines global system configurations, such as the module operation modes and the maximum message buffer configuration.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MDIS	FRZ	RFEN	HALT	NOTRDY	WAKMSK	SOFTRST	FRZACK	SUPV	SLFWAK	WRNEN	LPMACK	WAKSRC	Reserved	SRXDIS	IRMQ
W																
Reset	1	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DMA	PNET_EN	LPRIOEN	AEN	FDEN	0	IDAM		0	MAXMB						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

CAN_MCR field descriptions

Field	Description
0 MDIS	Module Disable This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN disables the clocks to the CAN Protocol Engine and Controller Host Interface sub-modules. This bit is not affected by soft reset.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	0 Enable the FlexCAN module. 1 Disable the FlexCAN module.
1 FRZ	Freeze Enable The FRZ bit specifies the FlexCAN behavior when the HALT bit in the CAN_MCR Register is set or when Debug mode is requested at chip level. When FRZ is asserted, FlexCAN is enabled to enter Freeze mode. Negation of this bit field causes FlexCAN to exit from Freeze mode. 0 Not enabled to enter Freeze mode. 1 Enabled to enter Freeze mode.
2 RFEN	Rx FIFO Enable This bit controls whether the Rx FIFO feature is enabled or not. When RFEN is set, MBs 0 to 5 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xDC) is used by the FIFO engine as well as additional MBs (up to 32, depending on CAN_CTRL2[RFFN] setting) which are used as Rx FIFO ID Filter Table elements. RFEN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in the table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (see Arbitration and matching timing). This bit can be written in Freeze mode only because it is blocked by hardware in other modes. **NOTE:** This bit cannot be set when CAN FD operation is enabled (see FDEN bit). 0 Rx FIFO not enabled. 1 Rx FIFO enabled.
3 HALT	Halt FlexCAN Assertion of this bit puts the FlexCAN module into Freeze mode. The CPU should clear it after initializing the Message Buffers and the Control Registers CAN_CTRL1 and CAN_CTRL2. No reception or transmission is performed by FlexCAN before this bit is cleared. Freeze mode cannot be entered while FlexCAN is in a low power mode. 0 No Freeze mode request. 1 Enters Freeze mode if the FRZ bit is asserted.
4 NOTRDY	FlexCAN Not Ready This read-only bit indicates that FlexCAN is either in Disable mode, Stop mode or Freeze mode. It is negated once FlexCAN has exited these modes. This bit is not affected by soft reset. 0 FlexCAN module is either in Normal mode, Listen-Only mode or Loop-Back mode. 1 FlexCAN module is either in Disable mode, Stop mode or Freeze mode.
5 WAKMSK	Wake Up Interrupt Mask This bit enables the Wake Up Interrupt generation under Self Wake Up mechanism. 0 Wake Up Interrupt is disabled. 1 Wake Up Interrupt is enabled.
6 SOFTRST	Soft Reset When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers.

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	The SOFTRST bit can be asserted directly by the CPU when it writes to the MCR Register. Because soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFTRST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in a low power mode. The module should be first removed from low power mode, and then soft reset can be applied. This bit is not affected by soft reset. 0 No reset request. 1 Resets the registers affected by soft reset.
7 FRZACK	Freeze Mode Acknowledge This read-only bit indicates that FlexCAN is in Freeze mode and its prescaler is stopped. The Freeze mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZACK bit to know when FlexCAN has actually entered Freeze mode. If Freeze Mode request is negated, then this bit is negated after the FlexCAN prescaler is running again. If Freeze mode is requested while FlexCAN is in a low power mode, then the FRZACK bit will be set only when the low-power mode is exited. See Section "Freeze Mode". This bit is not affected by soft reset. **NOTE:** FRZACK will be asserted within 178 CAN bits from the freeze mode request by the CPU, and negated within 2 CAN bits after the freeze mode request removal (see Section "Protocol Timing"). 0 FlexCAN not in Freeze mode, prescaler running. 1 FlexCAN in Freeze mode, prescaler stopped.
8 SUPV	Supervisor Mode This bit configures the FlexCAN to be either in Supervisor or User mode. The registers affected by this bit are marked as S/U in the Access Type column of the module memory map. Reset value of this bit is 1, so the affected registers start with Supervisor access allowance only. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 FlexCAN is in User mode. Affected registers allow both Supervisor and Unrestricted accesses. 1 FlexCAN is in Supervisor mode. Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location.
9 SLFWAK	Self Wake Up This bit enables the Self Wake Up feature when FlexCAN is in a low-power mode other than Disable mode. When this feature is enabled, the FlexCAN module monitors the bus for wake up event, that is, a recessive-to-dominant transition. If a wake up event is detected during Stop mode, then FlexCAN generates, if enabled to do so, a Wake Up interrupt to the CPU so that it can exit Stop mode globally and FlexCAN can request to resume the clocks. When FlexCAN is in a low-power mode other than Disable mode, this bit cannot be written as it is blocked by hardware. When Pretended Networking mode is set, this feature must be disabled. 0 FlexCAN Self Wake Up feature is disabled. 1 FlexCAN Self Wake Up feature is enabled.
10 WRNEN	Warning Interrupt Enable

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	When asserted, this bit enables the generation of the TWRNINT and RWRNINT flags in the Error and Status Register 1 (ESR1). If WRNEN is negated, the TWRNINT and RWRNINT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 TWRNINT and RWRNINT bits are zero, independent of the values in the error counters. 1 TWRNINT and RWRNINT bits are set when the respective error counter transitions from less than 96 to greater than or equal to 96.
11 LPMACK	Low-Power Mode Acknowledge This read-only bit indicates that FlexCAN is in a low-power mode (Disable mode, Stop mode). A low-power mode cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPMACK bit to know when FlexCAN has actually entered low power mode. This bit is not affected by soft reset. **NOTE:** LPMACK will be asserted within 180 CAN bits from the low-power mode request by the CPU, and negated within 2 CAN bits after the low-power mode request removal (see Section "Protocol Timing"). When FlexCAN is in Pretended Networking mode LPMACK will be negated within 180 CAN bits after the low-power mode request removal. 0 FlexCAN is not in a low-power mode. 1 FlexCAN is in a low-power mode.
12 WAKSRC	Wake Up Source This bit defines whether the integrated low-pass filter is applied to protect the Rx CAN input from spurious wake up. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 FlexCAN uses the unfiltered Rx input to detect recessive to dominant edges on the CAN bus. 1 FlexCAN uses the filtered Rx input to detect recessive to dominant edges on the CAN bus.
13 Reserved	This field is reserved. When writing to this field, always write the reset value.
14 SRXDIS	Self Reception Disable This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Self reception enabled. 1 Self reception disabled.
15 IRMQ	Individual Rx Masking And Queue Enable This bit indicates whether Rx matching process will be based either on individual masking and queue or on masking scheme with CAN_RXMGMASK, CAN_RX14MASK, CAN_RX15MASK and CAN_RXFGMASK. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Individual Rx masking and queue feature are disabled. For backward compatibility with legacy applications, the reading of C/S word locks the MB even if it is EMPTY. 1 Individual Rx masking and queue feature are enabled.
16 DMA	DMA Enable

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	The DMA Enable bit controls whether the DMA feature is enabled or not. The DMA feature can only be used in Rx FIFO, consequently the bit CAN_MCR[RFEN] must be asserted. When DMA and RFEN are set, the CAN_IFLAG1[BUF5I] generates the DMA request and no RX FIFO interrupt is generated. This bit can be written in Freeze mode only as it is blocked by hardware in other modes. 0 DMA feature for RX FIFO disabled. 1 DMA feature for RX FIFO enabled.
17 PNET_EN	Pretended Networking Enable This bit enables the Pretended Networking functionality. Once in Stop mode, CAN_PE sub-block is kept operational, able to process Rx message filtering as defined by the Pretended Networking configuration registers. See Receive Process under Pretended Networking Mode. This bit can be written in Freeze mode only. 0 Pretended Networking mode is disabled. 1 Pretended Networking mode is enabled.
18 LPRIOEN	Local Priority Enable This bit is provided for backwards compatibility with legacy applications. It controls whether the local priority feature is enabled or not. It is used to expand the ID used during the arbitration process. With this expanded ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Local Priority disabled. 1 Local Priority enabled.
19 AEN	Abort Enable When asserted, this bit enables the Tx abort mechanism. This mechanism guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. **NOTE:** When CAN_MCR[AEN] is asserted, only the abort mechanism (see Transmission abort mechanism) must be used for updating Mailboxes configured for transmission. **CAUTION:** Writing the Abort code into Rx Mailboxes can cause unpredictable results when the CAN_MCR[AEN] is asserted. 0 Abort disabled. 1 Abort enabled.
20 FDEN	CAN FD operation enable This bit enables the CAN with Flexible Data rate (CAN FD) operation. This bit can be written in Freeze mode only. **NOTE:** The Rx FIFO Enable (RFEN) bit cannot be set if FDEN is asserted. 1 CAN FD is enabled. FlexCAN is able to receive and transmit messages in both CAN FD and CAN 2.0 formats. 0 CAN FD is disabled. FlexCAN is able to receive and transmit messages in CAN 2.0 format.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 IDAM	ID Acceptance Mode

Table continues on the next page...

CAN_MCR field descriptions (continued)

Field	Description
	This 2-bit field identifies the format of the Rx FIFO ID Filter Table elements. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Section "Rx FIFO Structure". This field can be written only in Freeze mode because it is blocked by hardware in other modes. 00 Format A: One full ID (standard and extended) per ID Filter Table element. 01 Format B: Two full standard IDs or two partial 14-bit (standard and extended) IDs per ID Filter Table element. 10 Format C: Four partial 8-bit Standard IDs per ID Filter Table element. 11 Format D: All frames rejected.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 MAXMB	Number Of The Last Message Buffer This 7-bit field defines the number of the last Message Buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to a 16 MB configuration. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Number of the last MB = MAXMB **NOTE:** MAXMB must be programmed with a value smaller than or equal to the number of available Message Buffers, as described in FlexCAN Memory Partition for CAN FD. Additionally, the definition of MAXMB value must take into account the region of MBs occupied by Rx FIFO and its ID filters table space defined by RFFN bit in CAN_CTRL2 register. MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table "Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate" (see Arbitration and matching timing).

44.4.3 Control 1 register (CAN_CTRL1)

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back mode, Listen-Only mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler.

The CAN bit timing variables (PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW) can also be configured in CAN_CBT register, which extends the range of all these variables. If CAN_CBT[BTF] is set, PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW fields of CAN_CTRL1 become read only.

NOTE

When the CAN FD feature is enabled, do not use the PRESDIV, RJW, PSEG1, PSEG2, and PROPSEG fields of the CAN_CTRL1 register for CAN bit timing. Instead use the the CAN_CBT register's EPRESDIV, ERJW, EPSEG1, EPSEG2, and EPROPSEG fields.

The contents of this register are not affected by soft reset.

NOTE

The CAN bit variables in CAN_CTRL1 and in CAN_CBT are stored in the same register.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PRESDIV								RJW		PSEG1			PSEG2		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BOFFMSK	ERRMSK	CLKSRC	LPB	TWRNMSK	RWRNMSK	Reserved	Reserved	SMP	BOFFREC	TSYN	LBUF	LOM	PROPSEG		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CTRL1 field descriptions

Field	Description
0–7 PRESDIV	Prescaler Division Factor This 8-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The Maximum value of this field is 0xFF, that gives a minimum Sclock frequency equal to the PE clock frequency divided by 256. See Section "Protocol Timing". This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (PRESDIV + 1)
8–9 RJW	Resync Jump Width This 2-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization. One time quantum is equal to the Sclock period. The valid programmable values are 0–3. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = RJW + 1.
10–12 PSEG1	Phase Segment 1 This 3-bit field defines the length of Phase Segment 1 in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (PSEG1 + 1) × Time-Quanta.

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
13–15 PSEG2	Phase Segment 2 This 3-bit field defines the length of Phase Segment 2 in the bit time. The valid programmable values are 1–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 2 = (PSEG2 + 1) × Time-Quanta.
16 BOFFMSK	Bus Off Interrupt Mask This bit provides a mask for the Bus Off Interrupt BOFFINT in CAN_ESR1 register. 0 Bus Off interrupt disabled. 1 Bus Off interrupt enabled.
17 ERRMSK	Error Interrupt Mask This bit provides a mask for the Error Interrupt ERRINT in the CAN_ESR1 register. 0 Error interrupt disabled. 1 Error interrupt enabled.
18 CLKSRC	CAN Engine Clock Source This bit selects the clock source to the CAN Protocol Engine (PE) to be either the peripheral clock or the oscillator clock. The selected clock is the one fed to the prescaler to generate the Serial Clock (Sclock). In order to guarantee reliable operation, this bit can be written only in Disable mode because it is blocked by hardware in other modes. See Protocol timing". 0 The CAN engine clock source is the oscillator clock. Under this condition, the oscillator clock frequency must be lower than the bus clock. 1 The CAN engine clock source is the peripheral clock.
19 LPB	Loop Back Mode This bit configures FlexCAN to operate in Loop-Back mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. **NOTE:** In this mode, the CAN_MCR[SRXDIS] cannot be asserted because this will impede the self reception of a transmitted message. **NOTE:** The TDCEN bit in CAN_FDCTRL register must be disabled when LPB is asserted. 0 Loop Back disabled. 1 Loop Back enabled.
20 TWRNMSK	Tx Warning Interrupt Mask This bit provides a mask for the Tx Warning Interrupt associated with the TWRNINT flag in the Error and Status Register 1 (ESR1). This bit is read as zero when CAN_MCR[WRNEN] bit is negated. This bit can be written only if CAN_MCR[WRNEN] bit is asserted.

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	0 Tx Warning Interrupt disabled. 1 Tx Warning Interrupt enabled.
21 RWRNMSK	Rx Warning Interrupt Mask This bit provides a mask for the Rx Warning Interrupt associated with the RWRNINT flag in the Error and Status Register 1 (ESR1). This bit is read as zero when CAN_MCR[WRNEN] bit is negated. This bit can be written only if CAN_MCR[WRNEN] bit is asserted. 0 Rx Warning Interrupt disabled. 1 Rx Warning Interrupt enabled.
22 Reserved	This field is reserved.
23 Reserved	This field is reserved.
24 SMP	CAN Bit Sampling This bit defines the sampling mode of CAN bits at the Rx input. It can be written in Freeze mode only because it is blocked by hardware in other modes. **NOTE:** For proper operation, to assert SMP it is necessary to guarantee a minimum value of 2 TQs in CAN_CTRL1[PSEG1] (or CAN_CBT[EPSEG1]).This bit cannot be asserted when CAN FD is enabled (CAN_MCR[FDEN] = 1). 0 Just one sample is used to determine the bit value. 1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples; a majority rule is used.
25 BOFFREC	Bus Off Recovery This bit defines how FlexCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFFREC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FlexCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFFREC bit can be re-asserted again during Bus Off, but it will be effective only the next time the module enters Bus Off. If BOFFREC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery. 0 Automatic recovering from Bus Off state enabled. 1 Automatic recovering from Bus Off state disabled.
26 TSYN	Timer Sync This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special “SYNC” message, that is, global network time. If the RFEN bit in CAN_MCR is set (Rx FIFO enabled), the first available Mailbox, according to CAN_CTRL2[RFFN] setting, is used for timer synchronization instead of MB0. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Timer Sync feature disabled 1 Timer Sync feature enabled
27 LBUF	Lowest Buffer Transmitted First

Table continues on the next page...

CAN_CTRL1 field descriptions (continued)

Field	Description
	This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the CAN_MCR[LPRIOEN] bit does not affect the priority arbitration. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Buffer with highest priority is transmitted first. 1 Lowest number buffer is transmitted first.
28 LOM	Listen-Only Mode This bit configures FlexCAN to operate in Listen-Only mode. In this mode, transmission is disabled, all error counters described in CAN_ECR register are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error without changing the receive error counter (RXERRCNT) in CAN_ECR register, as if it was trying to acknowledge the message. Listen-Only mode is acknowledged by the state of CAN_ESR1[FLTCONF] field indicating Passive Error. There can be some delay between the Listen-Only mode request and acknowledge. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0 Listen-Only mode is deactivated. 1 FlexCAN module operates in Listen-Only mode.
29–31 PROPSEG	Propagation Segment This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = (PROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.

44.4.4 Free Running Timer (CAN_TIMER)

This register represents a 16-bit free running counter that can be read and written by the CPU. The timer starts from 0x0 after Reset, counts linearly to 0xFFFF, and wraps around.

When the TIMER_SRC in CAN_CTRL2 register is asserted, the timer is continuously incremented by an external time tick. The time tick must be synchronous to the Peripheral Clock, with a minimum pulse width of one clock cycle.

When the TIMER_SRC bit in CAN_CTRL2 register is negated, the timer is incremented by the CAN bit clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. The timer is not incremented during Disable, Stop, Pretended Networking and Freeze modes.

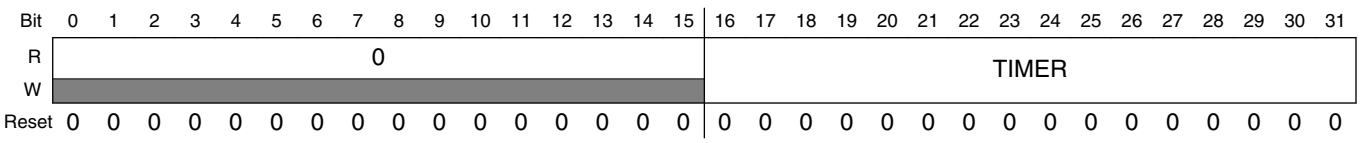
The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

If bit CAN_CTRL1[TSYN] is asserted, the Timer is reset whenever a message is received in the first available Mailbox, according to CAN_CTRL2[RFFN] setting.

The CPU can write to this register anytime. However, if the write occurs at the same time that the Timer is being reset by a reception in the first Mailbox, then the write value is discarded.

Reading this register affects the Mailbox Unlocking procedure, see Section "Mailbox Lock Mechanism".

Address: 0h base + 8h offset = 8h



CAN_TIMER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TIMER	Timer Value Contains the free-running counter value.

44.4.5 Rx Mailboxes Global Mask Register (CAN_RXMGMASK)

This register is located in RAM.

RXMGMASK is provided for legacy application support.

- When the CAN_MCR[IRMQ] bit is negated, RXMGMASK is always in effect (the bits in the MG field will mask the Mailbox filter bits).
- When the CAN_MCR[IRMQ] bit is asserted, RXMGMASK has no effect (the bits in the MG field will not mask the Mailbox filter bits).

RXMGMASK is used to mask the filter fields of all Rx MBs, excluding MBs 14-15, which have individual mask registers.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MG																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXMGMASK field descriptions

Field	Description
0–31 MG	Rx Mailboxes Global Mask Bits These bits mask the Mailbox filter bits. Note that the alignment with the ID word of the Mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE, which are located in the Control and Status word of the Mailbox. The following table shows in detail which MG bits mask each Mailbox filter field.

SMB[RTR] [1]	CAN_CTRL2[RRS]	CAN_CTRL2[EACEN]	Mailbox filter fields			
			MB[RTR]	MB[IDE]	MB[ID]	Reserved
0	-	0	note [2]	note [3]	MG[28:0]	MG[31:29]
0	-	1	MG[31]	MG[30]	MG[28:0]	MG[29]
1	0	-	-	-	-	MG[31:0]
1	1	0	-	-	MG[28:0]	MG[31:29]
1	1	1	MG[31]	MG[30]	MG[28:0]	MG[29]

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame.
3. If the CAN_CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame.

0 The corresponding bit in the filter is "don't care."
1 The corresponding bit in the filter is checked.

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If the CTRL2[EACEN] bit is negated, the RTR bit of Mailbox is never compared with the RTR bit of the incoming frame.
3. If the CAN_CTRL2[EACEN] bit is negated, the IDE bit of Mailbox is always compared with the IDE bit of the incoming frame.

44.4.6 Rx 14 Mask register (CAN_RX14MASK)

This register is located in RAM.

RX14MASK is provided for legacy application support. When the CAN_MCR[IRMQ] bit is asserted, RX14MASK has no effect.

RX14MASK is used to mask the filter fields of Message Buffer 14.

This register can only be programmed while the module is in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RX14M																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RX14MASK field descriptions

Field	Description
0–31 RX14M	Rx Buffer 14 Mask Bits Each mask bit masks the corresponding Mailbox 14 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

44.4.7 Rx 15 Mask register (CAN_RX15MASK)

This register is located in RAM.

RX15MASK is provided for legacy application support. When the CAN_MCR[IRMQ] bit is asserted, RX15MASK has no effect.

RX15MASK is used to mask the filter fields of Message Buffer 15.

This register can be programmed only while the module is in Freeze mode because it is blocked by hardware in other modes.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RX15M																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RX15MASK field descriptions

Field	Description
0–31 RX15M	Rx Buffer 15 Mask Bits

CAN_RX15MASK field descriptions (continued)

Field	Description
	Each mask bit masks the corresponding Mailbox 15 filter field in the same way that RXMGMASK masks other Mailboxes' filters. See the description of the CAN_RXMGMASK register. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

44.4.8 Error Counter (CAN_ECR)

This register has four 8-bit fields reflecting the value of the FlexCAN error counters:

- Transmit Error Counter (TXERRCNT field)
- Receive Error Counter (RXERRCNT field)
- Transmit Error Counter for errors detected in the Data Phase of CAN FD messages with the BRS bit set (TXERRCNT_FAST field)
- Receive Error Counter for errors detected in the Data Phase of CAN FD messages with the BRS bit set (RXERRCNT_FAST field)

The TXERRCNT and RXERRCNT counters take into account all errors in both CAN FD and non-FD message formats. TXERRCNT_FAST and RXERRCNT_FAST are dedicated to count only the errors occurred in the Data Phase of CAN FD frames with the BRS bit set.

The Fault Confinement State (FLTCONF field in Error and Status Register 1 - CAN_ESR1) is updated based on TXERRCNT and RXERRCNT counters only. TXERRCNT and RXERRCNT counters can be written in Freeze mode only. TXERRCNT_FAST and RXERRCNT_FAST counters are read-only except in Freeze mode where the CPU can write value zero. The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module.

The following are the basic rules for FlexCAN bus state transitions:

- If the value of TXERRCNT or RXERRCNT increases to be greater than or equal to 128, the FLTCONF field in the Error and Status Register is updated to reflect "Error Passive" state.
- If the FlexCAN state is "Error Passive", and either TXERRCNT or RXERRCNT decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLTCONF field in the Error and Status Register is updated to reflect "Error Active" state.

- If the value of TXERRCNT increases to be greater than 255, the FLTCONF field in the Error and Status Register is updated to reflect "Bus Off" state, and an interrupt may be issued. The value of TXERRCNT is then reset to zero.
- If FlexCAN is in "Bus Off' state, then TXERRCNT is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, TXERRCNT is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the TXERRCNT. When TXERRCNT reaches the value of 128, the FLTCONF field in the Error and Status Register is updated to be "Error Active" and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXERRCNT value. The TXERRCNT_FAST counter is frozen during busoff.
- If during system start-up, only one node is operating, then its TXERRCNT increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACKERR bit in the Error and Status Register). After the transition to "Error Passive" state, the TXERRCNT does not increment anymore by acknowledge errors. Therefore the device never goes to the "Bus Off" state.
- If the RXERRCNT increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to "Error Active" state.
- TXERRCNT_FAST and RXERRCNT_FAST error counters values increment and decrement based on errors detected only in the Data Phase of CAN FD frames with the BRS bit set, following the same increment and decrement rules as TXERRCNT and RXERRCNT counters. These counters do not wrap around and get stuck at their maximum value (255). They stop counting and keep their values frozen while FlexCAN is in "Bus Off" state. They are reset when FlexCAN leaves "Bus Off" state and restart counting once FlexCAN resumes to "Error Active" state.
- When FlexCAN is in Pretended Networking mode, the RXERRCNT and RXERRCNT_FAST keep counting errors and error flags are stored. The TXERRCNT and TXERRCNT_FAST preserve their values and do not change, since no transmission occurs under Pretended Networking mode. Error counters and error flags that changed values while in Pretended Networking mode are updated in CAN_ECR and CAN_ESR1 when FlexCAN resumes back to Normal mode. The FAST error flags in CAN_ESR1 register will not be set if FlexCAN is in Pretended Networking mode.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RXERRCNT_FAST								TXERRCNT_FAST								RXERRCNT								TXERRCNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ECR field descriptions

Field	Description
0–7 RXERRCNT_ FAST	Receive Error Counter for fast bits Receive Error Counter for errors detected in the Data Phase of received CAN FD messages with the BRS bit set. The RXERRCNT_FAST counter is read-only except in Freeze mode, where the CPU can write a 8-bit zero value only.
8–15 TXERRCNT_ FAST	Transmit Error Counter for fast bits Transmit Error Counter for errors detected in the Data Phase of transmitted CAN FD messages with the BRS bit set. The TXERRCNT_FAST counter is read-only except in Freeze mode, where the CPU can write a 8-bit zero value only.
16–23 RXERRCNT	Receive Error Counter Receive Error Counter for all errors detected in received messages. The RXERRCNT counter is read-only except in Freeze mode, where it can be written by the CPU.
24–31 TXERRCNT	Transmit Error Counter Transmit Error Counter for all errors detected in transmitted messages. The TXERRCNT counter is read-only except in Freeze mode, where it can be written by the CPU.

44.4.9 Error and Status 1 register (CAN_ESR1)

This register reports various error conditions detected in the reception and transmission of a CAN frame, some general status of the device and it is the source of some interrupts to the CPU.

The reported error conditions are BIT1ERR, BIT0ERR, ACKKERR, CRCERR, FRMERR and STFERR, for errors detected in CAN frames of any format, and BIT1ERR_FAST, BIT0ERR_FAST, CRCERR_FAST, FRMERR_FAST and STFERR_FAST for errors detected in the Data Phase of CAN FD frames with the BRS bit set only.

An error detected in a single CAN frame may be reported by one or more error flags. Also, error reporting is cummulative in case more error events happen in the next frames while the CPU does not attempt to read this register.

TXWRN, RXWRN, IDLE, TX, FLTCONF, RX and SYNCH are status bits.

BOFFINT, BOFFDONEINT, ERRINT, ERRINT_FAST, WAKINT, TWRNINT and RWRNINT are interrupt bits. It is recommended the CPU to use the following procedure when servicing interrupt requests generated by these bits:

- Read this register to capture all error condition and status bits. This action clear the respective bits that were set since the last read access.
- Write 1 to clear the interrupt bit that has triggered the interrupt request.
- Write 1 to clear the ERR_OVR bit if it is set.

Starting from all error flags cleared, a first error event sets either the ERRINT or the ERRINT_FAST (provided the corresponding mask bit is asserted). If other error events in subsequent frames happen before the CPU to serve the interrupt request, the ERR_OVR bit is set to indicate that errors from different frames had accumulated.

SYNCH	IDLE	TX	RX	FlexCAN State
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BIT1ERR_FAST	BIT0ERR_FAST	0	CRCERR_FAST	FRMERR_FAST	STFERR_FAST	0				ERROVR	ERRINT_FAST	BOFFDONEINT	SYNCH	TWRNINT	RWRNINT
W											w1c	w1c	w1c		w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BIT1ERR	BIT0ERR	ACKERR	CRCERR	FRMERR	STFERR	TXWRN	RXWRN	IDLE	TX	FLTCONF		RX	BOFFINT	ERRINT	WAKINT
W														w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ESR1 field descriptions

Field	Description
0 BIT1ERR_FAST	Bit1 Error in the Data Phase of CAN FD frames with the BRS bit set This bit indicates when an inconsistency occurs between the transmitted and the received bit in the Data Phase of CAN FD frames with the BRS bit set. 0 No such occurrence. 1 At least one bit sent as recessive is received as dominant.
1 BIT0ERR_FAST	Bit0 Error in the Data Phase of CAN FD frames with the BRS bit set This bit indicates when an inconsistency occurs between the transmitted and the received bit in the Data Phase of CAN FD frames with the BRS bit set. 0 No such occurrence. 1 At least one bit sent as dominant is received as recessive.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 CRCERR_FAST	Cyclic Redundancy Check Error in the CRC field of CAN FD frames with the BRS bit set This bit indicates that a CRC Error has been detected by the receiver node in the CRC field of CAN FD frames with the BRS bit set, that is, the calculated CRC is different from the received. 0 No such occurrence. 1 A CRC error occurred since last read of this register.
4 FRMERR_FAST	Form Error in the Data Phase of CAN FD frames with the BRS bit set This bit indicates that a Form Error has been detected by the receiver node in the Data Phase of CAN FD frames with the BRS bit set, that is, a fixed-form bit field contains at least one illegal bit.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
	0 No such occurrence. 1 A Form Error occurred since last read of this register.
5 STFERR_FAST	Stuffing Error in the Data Phase of CAN FD frames with the BRS bit set This bit indicates that a Stuffing Error has been detected in the Data Phase of CAN FD frames with the BRS bit set. 0 No such occurrence. 1 A Stuffing Error occurred since last read of this register.
6–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 ERROVR	Error Overrun bit This bit indicates that an error condition occurred when any error flag is already set. This bit is cleared by writing it to 1. 0 Overrun has not occurred. 1 Overrun has occurred.
11 ERRINT_FAST	Error Interrupt for errors detected in the Data Phase of CAN FD frames with the BRS bit set This bit indicates that at least one of the Error Bits detected in the Data Phase of CAN FD frames with the BRS bit set (BIT1ERR_FAST, BIT0ERR_FAST, CRCERR_FAST, FRMERR_FAST or STFERR_FAST) is set. If the corresponding mask bit CAN_CTRL2[ERRMSK_FAST] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence. 1 Indicates setting of any Error Bit detected in the Data Phase of CAN FD frames with the BRS bit set.
12 BOFFDONEINT	Bus Off Done Interrupt This bit is set when the Tx Error Counter (TXERRCNT) has finished counting 128 occurrences of 11 consecutive recessive bits on the CAN bus and is ready to leave Bus Off. If the corresponding mask bit in the Control 2 Register (BOFFDONEMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence. 1 FlexCAN module has completed Bus Off process.
13 SYNCH	CAN Synchronization Status This read-only flag indicates whether the FlexCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the FlexCAN. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not synchronized to the CAN bus. 1 FlexCAN is synchronized to the CAN bus.
14 TWRNINT	Tx Warning Interrupt Flag If the WRNEN bit in CAN_MCR is asserted, the TWRNINT bit is set when the TXWRN flag transitions from 0 to 1, meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control 1 Register (CAN_CTRL1[TWRNMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing 0 has no effect. This flag is not generated during Bus Off state. This bit is not updated during Freeze mode.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
	When FlexCAN returns to Normal mode from Pretended Network mode (see Receive Process under Pretended Networking Mode), this bit is not updated. 0 No such occurrence. 1 The Tx error counter transitioned from less than 96 to greater than or equal to 96.
15 RWRNINT	Rx Warning Interrupt Flag If the WRNEN bit in CAN_MCR is asserted, the RWRNINT bit is set when the RXWRN flag transitions from 0 to 1, meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control 1 Register (CAN_CTRL1[RWRNMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when the WRNEN is set again. Writing 0 has no effect. This bit is not updated during Freeze mode. When FlexCAN returns to Normal mode from Pretended Network mode (see Receive Process under Pretended Networking Mode), this bit is updated to reflect the Rx error counter state. 0 No such occurrence. 1 The Rx error counter transitioned from less than 96 to greater than or equal to 96.
16 BIT1ERR	Bit1 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message. This bit is updated when FlexCAN returns to Normal mode from Pretended Network mode. **NOTE:** This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits. 0 No such occurrence. 1 At least one bit sent as recessive is received as dominant.
17 BIT0ERR	Bit0 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message. This bit is updated when FlexCAN returns to Normal mode from Pretended Network mode. 0 No such occurrence. 1 At least one bit sent as dominant is received as recessive.
18 ACKERR	Acknowledge Error This bit indicates that an Acknowledge Error has been detected by the transmitter node, that is, a dominant bit has not been detected during the ACK SLOT. This bit is updated when FlexCAN returns to Normal mode from Pretended Network mode. 0 No such occurrence. 1 An ACK error occurred since last read of this register.
19 CRCERR	Cyclic Redundancy Check Error This bit indicates that a CRC Error has been detected by the receiver node either in a non-FD message or in the arbitration or data phase of a frame in CAN FD format, that is, the calculated CRC is different from the received. This bit is updated when FlexCAN returns to Normal mode from Pretended Network mode.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
	0 No such occurrence. 1 A CRC error occurred since last read of this register.
20 FRMERR	Form Error This bit indicates that a Form Error has been detected in a non-FD message or else in an FD message's arbitration or data phase by the receiver node, that is, a fixed-form bit field contains at least one illegal bit. This bit is updated when FlexCAN returns to Normal mode from Pretended Network mode. 0 No such occurrence. 1 A Form Error occurred since last read of this register.
21 STFERR	Stuffing Error This bit indicates that a Stuffing Error has been detected in a non-FD message or else in an FD message's arbitration or data phase by the receiver node. This bit is updated when FlexCAN returns to Normal mode from Pretended Network mode. 0 No such occurrence. 1 A Stuffing Error occurred since last read of this register.
22 TXWRN	TX Error Warning This bit indicates when repetitive errors are occurring during message transmission and is affected by the value of TXERRCNT in CAN_ECR register only. This bit is not updated during Freeze mode. 0 No such occurrence. 1 TXERRCNT is greater than or equal to 96.
23 RXWRN	Rx Error Warning This bit indicates when repetitive errors are occurring during message reception and is affected by the value of RXERRCNT in CAN_ECR register only. This bit is not updated during Freeze mode. Additionally, it is updated when FlexCAN returns to Normal mode from Pretended Networking mode. 0 No such occurrence. 1 RXERRCNT is greater than or equal to 96.
24 IDLE	This bit indicates when CAN bus is in IDLE state. See the table in the overall CAN_ESR1 register description. 0 No such occurrence. 1 CAN bus is now IDLE.
25 TX	FlexCAN In Transmission This bit indicates if FlexCAN is transmitting a message. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not transmitting a message. 1 FlexCAN is transmitting a message.
26–27 FLTCONF	Fault Confinement State This 2-bit field indicates the Confinement State of the FlexCAN module.

Table continues on the next page...

CAN_ESR1 field descriptions (continued)

Field	Description
	If the LOM bit in the Control Register 1 is asserted, after some delay that depends on the CAN bit timing the FLTCONF field will indicate “Error Passive”. The very same delay affects the way how FLTCONF reflects an update to CAN_ECR register by the CPU. It may be necessary up to one CAN bit time to get them coherent again. This bit field is affected by soft reset, but if the LOM bit is asserted, its reset value lasts just one CAN bit. After this time, FLTCONF reports "Error Passive". 00 Error Active 01 Error Passive 1x Bus Off
28 RX	FlexCAN In Reception This bit indicates if FlexCAN is receiving a message. See the table in the overall CAN_ESR1 register description. 0 FlexCAN is not receiving a message. 1 FlexCAN is receiving a message.
29 BOFFINT	Bus Off Interrupt This bit is set when FlexCAN enters ‘Bus Off’ state. If the corresponding mask bit in the Control Register 1 (CAN_CTRL1[BOFFMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence. 1 FlexCAN module entered Bus Off state.
30 ERRINT	Error Interrupt This bit indicates that at least one of the Error Bits (BIT1ERR, BIT0ERR, ACKERR. CRCERR, FRMERR or STFERR) is set. If the corresponding mask bit CAN_CTRL1[ERRMSK] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0 No such occurrence. 1 Indicates setting of any Error Bit in the Error and Status Register.
31 WAKINT	Wake-Up Interrupt This field applies when FlexCAN is in low-power mode under Self Wake Up mechanism: • Stop mode When a recessive-to-dominant transition is detected on the CAN bus and if the CAN_MCR[WAKMSK] bit is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When CAN_MCR[SLFWAK] is negated, this flag is masked. The CPU must clear this flag before disabling the bit. Otherwise it will be set when the SLFWAK is set again. Writing 0 has no effect. 0 No such occurrence. 1 Indicates a recessive to dominant transition was received on the CAN bus.

44.4.10 Interrupt Masks 2 register (CAN_IMASK2)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB63 to MB32. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding CAN_IFLAG2 bit is set.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUF63TO32M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IMASK2 field descriptions

Field	Description
0–31 BUF63TO32M	Buffer MB $_i$ Mask Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB63 to MB32. **NOTE:** Setting or clearing a bit in the CAN_IMASK2 Register can assert or negate an interrupt request, if the corresponding IFLAG2 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

44.4.11 Interrupt Masks 1 register (CAN_IMASK1)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB31 to MB0. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding CAN_IFLAG1 bit is set.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUF31TO0M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IMASK1 field descriptions

Field	Description
0–31 BUF31TO0M	Buffer MB $_i$ Mask

CAN_IMASK1 field descriptions (continued)

Field	Description
	Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB31 to MB0. **NOTE:** Setting or clearing a bit in the CAN_IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

44.4.12 Interrupt Flags 2 register (CAN_IFLAG2)

This register defines the flags for the 32 Message Buffer interrupts for MB63 to MB32. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the respective CAN_IFLAG2 bit. If the corresponding CAN_IMASK2 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

Before updating CAN_MCR[MAXMB] field, CPU must service the CAN_IFLAG2 bits whose MB value is greater than the MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BUF63TO32I																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IFLAG2 field descriptions

Field	Description
0–31 BUF63TO32I	Buffer MB $_i$ Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB63 to MB32. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception. 1 The corresponding buffer has successfully completed transmission or reception.

44.4.13 Interrupt Flags 1 register (CAN_IFLAG1)

This register defines the flags for the 32 Message Buffer interrupts for MB31 to MB0. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding CAN_IFLAG1 bit. If the corresponding CAN_IMASK1 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect. There is an exception when DMA for Rx FIFO is enabled, as described below.

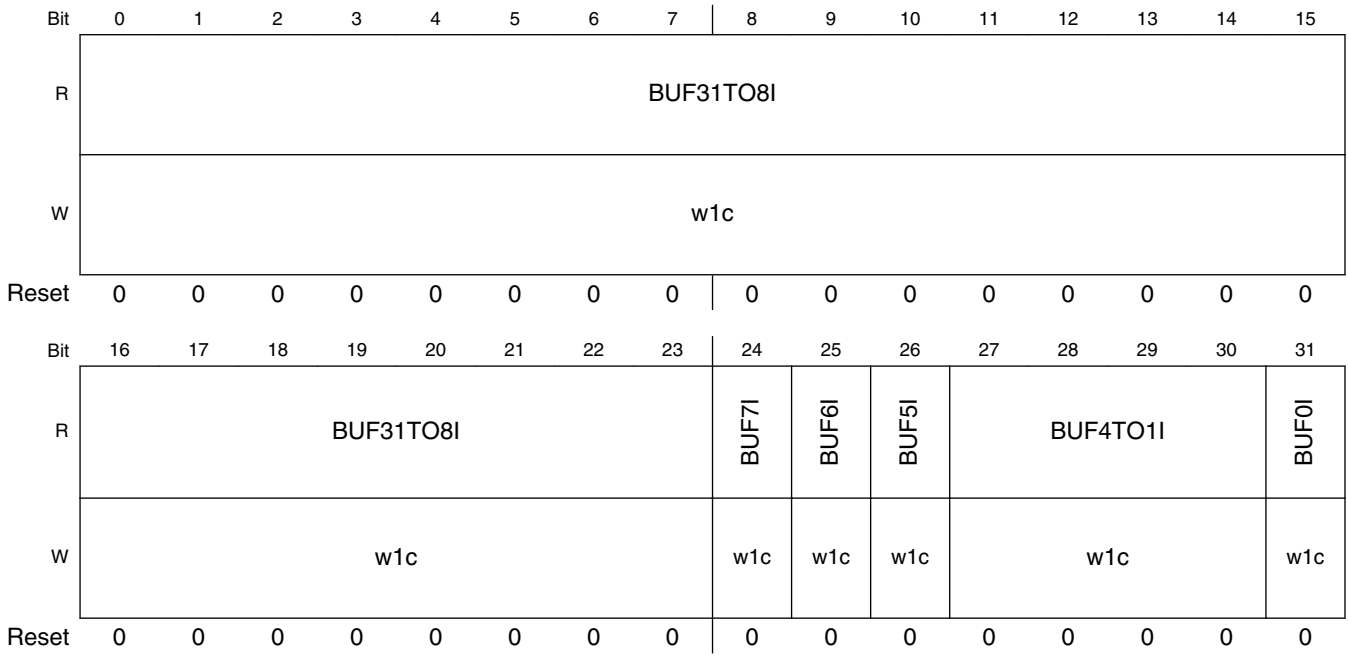
The BUF7I to BUF5I flags are also used to represent FIFO interrupts when the Rx FIFO is enabled. When the bit CAN_MCR[RFEN] is set and the bit CAN_MCR[DMA] is negated, the function of the 8 least significant interrupt flags changes: BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO, BUF0I is used to empty FIFO, and BUF4I to BUF1I bits are reserved.

Before enabling the CAN_MCR[RFEN], the CPU must service the IFLAG bits asserted in the Rx FIFO region; see Section "Rx FIFO". Otherwise, these IFLAG bits will mistakenly show the related MBs now belonging to FIFO as having contents to be serviced. When the CAN_MCR[RFEN] bit is negated, the FIFO flags must be cleared. The same care must be taken when an CAN_CTRL2[RFFN] value is selected extending Rx FIFO filters beyond MB7. For example, when RFFN is 0x8, the MB0-23 range is occupied by Rx FIFO filters and related IFLAG bits must be cleared.

When both the CAN_MCR[RFEN] and CAN_MCR[DMA] bits are asserted (DMA feature for Rx FIFO enabled), the function of the 8 least significant interrupt flags (BUF7I - BUF0I) are changed to support the DMA operation. BUF7I and BUF6I are not used, as well as, BUF4I to BUF1I. BUF5I indicates operating condition of FIFO, and BUF0I is used to empty FIFO. Moreover, BUF5I does not generate a CPU interrupt, but generates a DMA request. IMASK1 bits in Rx FIFO region are not considered when bit CAN_MCR[DMA] is enabled. In addition the CPU must not clear the flag BUF5I when DMA is enabled. Before enabling the bit CAN_MCR[DMA], the CPU must service the IFLAGs asserted in the Rx FIFO region. When the bit CAN_MCR[DMA] is negated, the FIFO must be empty. FIFO must be disabled when FDEN bit in CAN_MCR register is enabled.

Before updating CAN_MCR[MAXMB] field, CPU must service the CAN_IFLAG1 bits whose MB value is greater than the CAN_MCR[MAXMB] to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 0h base + 30h offset = 30h



CAN_IFLAG1 field descriptions

Field	Description
0–23 BUF31TO8I	Buffer MB_i Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB31 to MB8. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception. 1 The corresponding buffer has successfully completed transmission or reception.
24 BUF7I	Buffer MB7 Interrupt Or "Rx FIFO Overflow" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), this bit flags the interrupt for MB7. **NOTE:** This flag is cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes. The BUF7I flag represents "Rx FIFO Overflow" when CAN_MCR[RFEN] is set. In this case, the flag indicates that a message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. 0 No occurrence of MB7 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO overflow when MCR[RFEN]=1 1 MB7 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO overflow when MCR[RFEN]=1
25 BUF6I	Buffer MB6 Interrupt Or "Rx FIFO Warning" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), this bit flags the interrupt for MB6. **NOTE:** This flag is cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes.

Table continues on the next page...

CAN_IFLAG1 field descriptions (continued)

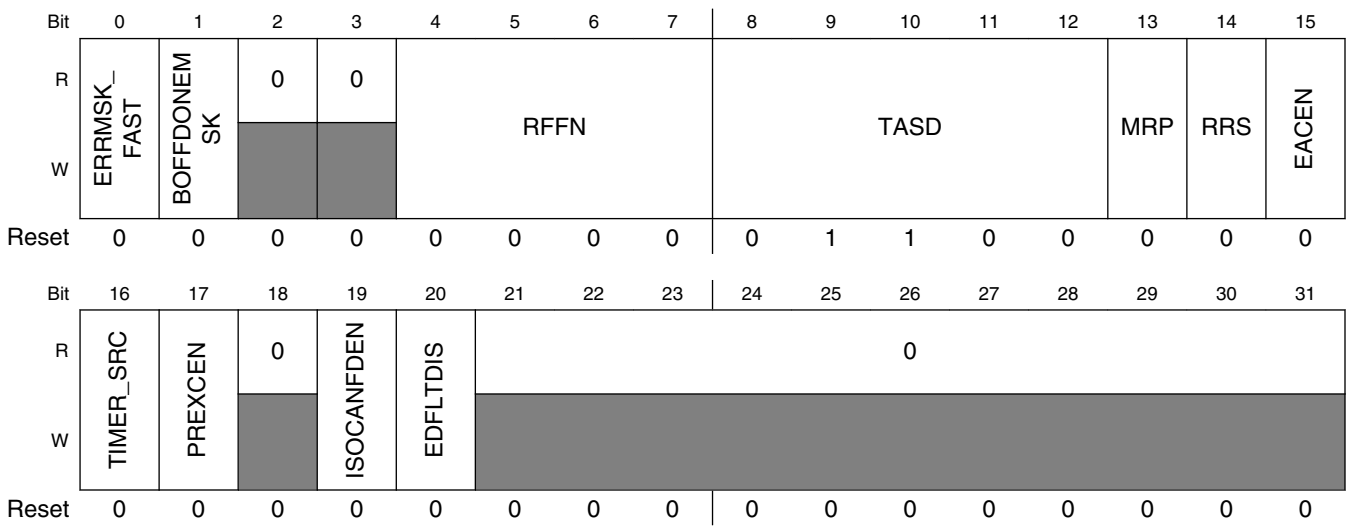
Field	Description
	The BUF6I flag represents "Rx FIFO Warning" when CAN_MCR[RFEN] is set. In this case, the flag indicates when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. Note that if the flag is cleared while the number of unread messages is greater than 4, it does not assert again until the number of unread messages within the Rx FIFO is decreased to be equal to or less than 4. 0 No occurrence of MB6 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO almost full when MCR[RFEN]=1 1 MB6 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO almost full when MCR[RFEN]=1
26 BUF5I	Buffer MB5 Interrupt Or "Frames available in Rx FIFO" When the RFEN bit in the MCR is cleared (Rx FIFO disabled), this bit flags the interrupt for MB5. **NOTE:** This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by CPU writes. When MCR[RFEN] is set (Rx FIFO enabled), the BUF5I flag represents "Frames available in Rx FIFO" and indicates that at least one frame is available to be read from the Rx FIFO. When the MCR[DMA] bit is enabled, this flag generates a DMA request and the CPU must not clear this bit by writing 1 in BUF5I. 0 No occurrence of MB5 completing transmission/reception when MCR[RFEN]=0, or of frame(s) available in the FIFO, when MCR[RFEN]=1 1 MB5 completed transmission/reception when MCR[RFEN]=0, or frame(s) available in the Rx FIFO when MCR[RFEN]=1. It generates a DMA request in case of MCR[RFEN] and MCR[DMA] are enabled.
27–30 BUF4TO1I	Buffer MB $_i$ Interrupt Or "reserved" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), these bits flag the interrupts for MB4 to MB1. **NOTE:** These flags are cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes. The BUF4TO1I flags are reserved when CAN_MCR[RFEN] is set. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0. 1 The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0.
31 BUF0I	Buffer MB0 Interrupt Or "reserved" When the RFEN bit in the CAN_MCR register is cleared (Rx FIFO disabled), this bit flags the interrupt for MB0. **NOTE:** This flag is cleared by the FlexCAN whenever the bit CAN_MCR[RFEN] is changed by CPU writes. The BUF0I flag is reserved when CAN_MCR[RFEN] is set. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0. 1 The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0.

44.4.14 Control 2 register (CAN_CTRL2)

This register complements Control1 Register providing control bits for memory write access in Freeze Mode, for extending FIFO filter quantity, and for adjust the operation of internal FlexCAN processes like matching and arbitration.

The contents of this register are not affected by soft reset.

Address: 0h base + 34h offset = 34h



CAN_CTRL2 field descriptions

Field	Description
0 ERRMSK_FAST	Error Interrupt Mask for errors detected in the Data Phase of fast CAN FD frames This bit provides a mask for the ERRINT_FAST Interrupt in CAN_ESR1 register. 0 ERRINT_FAST Error interrupt disabled. 1 ERRINT_FAST Error interrupt enabled.
1 BOFFDONEMSK	Bus Off Done Interrupt Mask This bit provides a mask for the Bus Off Done Interrupt in CAN_ESR1 register. 0 Bus Off Done interrupt disabled. 1 Bus Off Done interrupt enabled.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 RFFN	Number Of Rx FIFO Filters This 4-bit field defines the number of Rx FIFO filters, as shown in the following table. The maximum selectable number of filters is determined by the chip. This field can only be written in Freeze mode as it is blocked by hardware in other modes. This field must not be programmed with values that make the

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

Field	Description
	number of Message Buffers occupied by Rx FIFO and ID Filter exceed the number of Mailboxes present, defined by CAN_MCR[MAXMB]. **NOTE:** Each group of eight filters occupies a memory space equivalent to two Message Buffers which means that the more filters are implemented the less Mailboxes will be available. Considering that the Rx FIFO occupies the memory space originally reserved for MB0-5, RFFN should be programmed with a value corresponding to a number of filters not greater than the number of available memory words which can be calculated as follows: (SETUP_MB - 6) × 4 where SETUP_MB is the least between the parameter NUMBER_OF_MB and CAN_MCR[MAXMB]. The number of remaining Mailboxes available will be: (SETUP_MB - 8) - (RFFN × 2) If the Number of Rx FIFO Filters programmed through RFFN exceeds the SETUP_MB value (memory space available) the exceeding ones will not be functional. **NOTE:** • The number of the last remaining available mailboxes is defined by the least value between the NUMBER_OF_MB minus 1 and the CAN_MCR[MAXMB] field. • If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.
8–12 TASD	Tx Arbitration Start Delay

RFFN[3:0]	Number of Rx FIFO filter elements	Message Buffers occupied by Rx FIFO and ID Filter Table	Remaining Available Mailboxes	Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks	Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask
0x0	8	MB 0-7	MB 8-95	Elements 0-7	none
0x1	16	MB 0-9	MB 10-95	Elements 0-9	Elements 10-15
0x2	24	MB 0-11	MB 12-95	Elements 0-11	Elements 12-23
0x3	32	MB 0-13	MB 14-95	Elements 0-13	Elements 14-31
0x4	40	MB 0-15	MB 16-95	Elements 0-15	Elements 16-39
0x5	48	MB 0-17	MB 18-95	Elements 0-17	Elements 18-47
0x6	56	MB 0-19	MB 20-95	Elements 0-19	Elements 20-55
0x7	64	MB 0-21	MB 22-95	Elements 0-21	Elements 22-63
0x8	72	MB 0-23	MB 24-95	Elements 0-23	Elements 24-71
0x9	80	MB 0-25	MB 26-95	Elements 0-25	Elements 26-79
0xA	88	MB 0-27	MB 28-95	Elements 0-27	Elements 28-87
0xB	96	MB 0-29	MB 30-95	Elements 0-29	Elements 30-95
0xC	104	MB 0-31	MB 32-95	Elements 0-31	Elements 32-103
0xD	112	MB 0-33	MB 34-95	Elements 0-31	Elements 32-111
0xE	120	MB 0-35	MB 36-95	Elements 0-31	Elements 32-119
0xF	128	MB 0-37	MB 38-95	Elements 0-31	Elements 32-127

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

Field	Description
	This 5-bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on CAN bus. See Tx Arbitration start delay for more details. This field can be written only in Freeze mode because it is blocked by hardware in other modes.
13 MRP	Mailboxes Reception Priority If this bit is set the matching process starts from the Mailboxes and if no match occurs the matching continues on the Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Matching starts from Rx FIFO and continues on Mailboxes. 1 Matching starts from Mailboxes and continues on Rx FIFO.
14 RRS	Remote Request Storing If this bit is asserted Remote Request Frame is submitted to a matching process and stored in the corresponding Message Buffer in the same fashion of a Data Frame. No automatic Remote Response Frame will be generated. If this bit is negated the Remote Request Frame is submitted to a matching process and an automatic Remote Response Frame is generated if a Message Buffer with CODE=0b1010 is found with the same ID. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Remote Response Frame is generated. 1 Remote Request Frame is stored.
15 EACEN	Entire Frame Arbitration Field Comparison Enable For Rx Mailboxes This bit controls the comparison of IDE and RTR bits within Rx Mailboxes filters with their corresponding bits in the incoming frame by the matching process. This bit does not affect matching for Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0 Rx Mailbox filter's IDE bit is always compared and RTR is never compared despite mask bits. 1 Enables the comparison of both Rx Mailbox filter's IDE and RTR bit with their corresponding bits within the incoming frame. Mask bits do apply.
16 TIMER_SRC	Timer Source Selects the time tick source used for incrementing the Free Running Timer counter. This bit can be written in Freeze mode only. 0 The Free Running Timer is clocked by the CAN bit clock, which defines the baud rate on the CAN bus. 1 The Free Running Timer is clocked by an external time tick. The period can be either adjusted to be equal to the baud rate on the CAN bus, or a different value as required. See the device specific section for details about the external time tick.
17 PREXCEN	Protocol Exception Enable This bit enables the Protocol Exception feature. This field is writable only in Freeze mode. 0 Protocol Exception is disabled. 1 Protocol Exception is enabled.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 ISOCANFDEN	ISO CAN FD Enable

Table continues on the next page...

CAN_CTRL2 field descriptions (continued)

Field	Description
	This field enables the CAN FD protocol according to ISO specification (ISO 11898-1) (see CAN FD ISO compliance). This field is writable only in Freeze mode. 0 FlexCAN operates using the non-ISO CAN FD protocol. 1 FlexCAN operates using the ISO CAN FD protocol (ISO 11898-1).
20 EDFLTDIS	Edge Filter Disable This bit disables the Edge Filter used during the bus integration state. When the Edge Filter is enabled, two consecutive nominal time quanta with dominant bus state are required to detect an edge that causes synchronization. When synchronization occurs, the counting of the sequence of eleven consecutive recessive bits is restarted. The Edge Filter prevents the dominant pulses that are shorter than a nominal bit time (present during the data phase of an FD Frame) from being mistaken for an idle condition. This field is writable only in Freeze mode. 0 Edge Filter is enabled. 1 Edge Filter is disabled.
21–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

44.4.15 Error and Status 2 register (CAN_ESR2)

This register reports some general status information.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0									LPTM						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	VPS	IMB	0												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_ESR2 field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 LPTM	Lowest Priority Tx Mailbox If CAN_ESR2[VPS] is asserted, this field indicates the lowest number inactive Mailbox (see the CAN_ESR2[IMB] bit description). If there is no inactive Mailbox then the Mailbox indicated depends on CAN_CTRL1[LBUF] bit value. If CAN_CTRL1[LBUF] bit is negated then the Mailbox indicated is the one

Table continues on the next page...

CAN_ESR2 field descriptions (continued)

Field	Description
	that has the greatest arbitration value (see the "Highest priority Mailbox first" section). If CAN_CTRL1[LBUF] bit is asserted then the Mailbox indicated is the highest number active Tx Mailbox. If a Tx Mailbox is being transmitted it is not considered in LPTM calculation. If CAN_ESR2[IMB] is not asserted and a frame is transmitted successfully, LPTM is updated with its Mailbox number.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 VPS	Valid Priority Status This bit indicates whether CAN_ESR2[IMB] and CAN_ESR2[LPTM] contents are currently valid or not. It is asserted upon every complete Tx arbitration process unless the CPU writes to Control and Status word of a Mailbox that has already been scanned, that is, it is behind Tx Arbitration Pointer, during the Tx arbitration process. If there is no inactive Mailbox and only one Tx Mailbox that is being transmitted then VPS is not asserted. This bit is negated upon the start of every Tx arbitration process or upon a write to Control and Status word of any Mailbox. **NOTE:** CAN_ESR2[VPS] is not affected by any CPU write into Control Status (C/S) of a MB that is blocked by abort mechanism. When CAN_MCR[AEN] is asserted, the abort code write in C/S of a MB that is being transmitted (pending abort), or any write attempt into a Tx MB with CAN_IFLAG set is blocked. 0 Contents of IMB and LPTM are invalid. 1 Contents of IMB and LPTM are valid.
18 IMB	Inactive Mailbox If ESR2[VPS] is asserted, this bit indicates whether there is any inactive Mailbox (CODE field is either 0b1000 or 0b0000). This bit is asserted in the following cases: • During arbitration, if an CAN_ESR2[LPTM] is found and it is inactive. • If CAN_ESR2[IMB] is not asserted and a frame is transmitted successfully. This bit is cleared in all start of arbitration (see Section "Arbitration process"). **NOTE:** CAN_ESR2[LPTM] mechanism have the following behavior: if an MB is successfully transmitted and CAN_ESR2[IMB]=0 (no inactive Mailbox), then CAN_ESR2[VPS] and CAN_ESR2[IMB] are asserted and the index related to the MB just transmitted is loaded into CAN_ESR2[LPTM]. 0 If ESR2[VPS] is asserted, the ESR2[LPTM] is not an inactive Mailbox. 1 If ESR2[VPS] is asserted, there is at least one inactive Mailbox. LPTM content is the number of the first one.
19–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

44.4.16 CRC Register (CAN_CRCR)

This register provides information about the CRC of transmitted messages for non FD messages. This register only reports the 15 low order bits of CRC calculations for messages in CAN FD format that require either 17 or 21 bits. For CAN FD format frames, the CAN_FDCRC register must be used. This register is updated at the same time the Tx Interrupt Flag is asserted.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0									MBCRC						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	TXCRC														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CRCR field descriptions

Field	Description
0–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MBCRC	CRC Mailbox This field indicates the number of the Mailbox corresponding to the value in CAN_CRCR[TXCRC] field.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17–31 TXCRC	Transmitted CRC value This field indicates the CRC value of the last transmitted message for non-FD frames. For FD frames, CRC value is reported in CAN_FDCRC register.

44.4.17 Rx FIFO Global Mask register (CAN_RXFGMASK)

This register is located in RAM.

If Rx FIFO is enabled, RXFGMASK is used to mask the Rx FIFO ID Filter Table elements that do not have a corresponding RXIMR according to CAN_CTRL2[RFFN] field setting.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	FGM																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXFGMASK field descriptions

Field	Description
0–31 FGM	Rx FIFO Global Mask Bits These bits mask the ID Filter Table elements bits in a perfect alignment. The following table shows how the FGM bits correspond to each IDAF field.

Rx FIFO ID Filter Table Elements Format (CAN_MCR[IDAM])	Identifier Acceptance Filter Fields					
	RTR	IDE	RXIDA	RXIDB [1]	RXIDC [2]	Reserved
A	FGM[31]	FGM[30]	FGM[29:1]	-	-	FGM[0]
B	FGM[31], FGM[15]	FGM[30], FGM[14]	-	FGM[29:16], FGM[13:0]		-
C	-	-		-	FGM[31:24], FGM[23:16], FGM[15:8], FGM[7:0]	

1. If CAN_MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If CAN_MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

0 The corresponding bit in the filter is "don't care."
1 The corresponding bit in the filter is checked.

1. If CAN_MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If CAN_MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

44.4.18 Rx FIFO Information Register (CAN_RXFIR)

RXFIR provides information on Rx FIFO.

This register is the port through which the CPU accesses the output of the RXFIR FIFO located in RAM. The RXFIR FIFO is written by the FlexCAN whenever a new message is moved into the Rx FIFO as well as its output is updated whenever the output of the Rx FIFO is updated with the next message. See Section "Rx FIFO" for instructions on reading this register.

Address: 0h base + 4Ch offset = 4Ch

Bit	0–22	23–31
R	0	IDHIT
W		
Reset	x*	x*

* Notes:
- x = Undefined at reset.

CAN_RXFIR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 IDHIT	Identifier Acceptance Filter Hit Indicator This field indicates which Identifier Acceptance Filter was hit by the received message that is in the output of the Rx FIFO. If multiple filters match the incoming message ID then the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only while the CAN_IFLAG1[BUF5I] is asserted.

44.4.19 CAN Bit Timing Register (CAN_CBT)

This register is an alternative way to store the CAN bit timing variables described in CAN_CTRL1 register. EPRESDIV, EPROPSEG, EPSEG1, EPSEG2 and ERJW are extended versions of PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW bit fields respectively.

The BTF bit selects the use of the timing variables defined in this register.

The contents of this register are not affected by soft reset.

NOTE

The CAN bit variables in CAN_CTRL1 and in CAN_CBT are stored in the same register.

NOTE

When the CAN FD feature is enabled (CAN_MCR[FDEN] is set), always set CAN_CBT[BTF].

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	BTF	EPRESDIV										ERJW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EPROPSEG						EPSEG1					EPSEG2				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CBT field descriptions

Field	Description
0 BTF	Bit Timing Format Enable Enables the use of extended CAN bit timing fields EPRESDIV, EPROPSEG, EPSEG1, EPSEG2 and ERJW replacing the CAN bit timing variables defined in CAN_CTRL1 register. This field can be written in Freeze mode only. 0 Extended bit time definitions disabled. 1 Extended bit time definitions enabled.
1–10 EPRESDIV	Extended Prescaler Division Factor This 10-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PRESDIV] value range. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency (see Protocol timing). This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (EPRESDIV + 1)
11–15 ERJW	Extended Resync Jump Width This 5-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[RJW] value range. One time quantum is equal to the Sclock period. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = ERJW + 1.
16–21 EPROPSEG	Extended Propagation Segment This 6-bit field defines the length of the Propagation Segment in the bit time when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PROPSEG] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = (EPROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.
22–26 EPSEG1	Extended Phase Segment 1 This 5-bit field defines the length of Phase Segment 1 in the bit time when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PSEG1] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (EPSEG1 + 1) × Time-Quanta. Time-Quantum = one Sclock period.

Table continues on the next page...

CAN_CBT field descriptions (continued)

Field	Description
27–31 EPSEG2	Extended Phase Segment 2 This 5-bit field defines the length of Phase Segment 2 in the bit time when CAN_CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CAN_CTRL1[PSEG2] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (EPSEG2 + 1) × Time-Quanta. Time-Quantum = one Sclock period.

44.4.20 Interrupt Masks 3 Register (CAN_IMASK3)

This register allows any number of a range of the 32 Message Buffer Interrupts to be enabled or disabled for MB95 to MB64. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding CAN_IFLAG3 bit is set.

Address: 0h base + 6Ch offset = 6Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	BUF95TO64M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IMASK3 field descriptions

Field	Description
0–31 BUF95TO64M	Buffer MB $_i$ Mask Each bit enables or disables the corresponding FlexCAN Message Buffer Interrupt for MB95 to MB64. When CAN FD is enabled, the MB range is defined in accordance to the MBDSRs bit fields of CAN_FDCTRL register. **NOTE:** Setting or clearing a bit in the CAN_IMASK3 Register can assert or negate an interrupt request, if the corresponding CAN_IFLAG3 bit is set. 0 The corresponding buffer Interrupt is disabled. 1 The corresponding buffer Interrupt is enabled.

44.4.21 Interrupt Flags 3 Register (CAN_IFLAG3)

This register defines the flags for the 32 Message Buffer interrupts for MB95 to MB64. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding CAN_IFLAG3 bit. If the corresponding CAN_IMASK3 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

Before updating CAN_MCR[MAXMB] field, CPU must service the CAN_IFLAG3 bits whose MB value is greater than the MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	BUF95TO64																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_IFLAG3 field descriptions

Field	Description
0–31 BUF95TO64	Buffer MB $_i$ Interrupt Each bit flags the corresponding FlexCAN Message Buffer interrupt for MB95 to MB64.When CAN FD is enabled, the MB range is defined in accordance to the MBDSRs bit fields of CAN_FDCTRL register. 0 The corresponding buffer has no occurrence of successfully completed transmission or reception 1 The corresponding buffer has successfully completed transmission or reception

44.4.22 Rx Individual Mask Registers (CAN_RXIMR*n*)

The RX Individual Mask Registers are used to store the acceptance masks for ID filtering in Rx MBs and the Rx FIFO.

When the Rx FIFO is disabled (CAN_MCR[RFEN] bit is negated), an individual mask is provided for each available Rx Mailbox on a one-to-one correspondence. When the Rx FIFO is enabled (CAN_MCR[RFEN] bit is asserted), an individual mask is provided for each Rx FIFO ID Filter Table Element on a one-to-one correspondence depending on the setting of CAN_CTRL2[RFFN] (see Rx FIFO).

CAN_RXIMR0 stores the individual mask associated to either MB0 or ID Filter Table Element 0, CAN_RXIMR1 stores the individual mask associated to either MB1 or ID Filter Table Element 1 and so on.

CAN_RXIMR registers can only be accessed by the CPU while the module is in Freeze mode, otherwise, they are blocked by hardware. These registers are not affected by reset. They are located in RAM and must be explicitly initialized prior to any reception.

It is possible for the RXIMR memory region to be accessed as general purpose memory. See Bus interface for more information.

Address: 0h base + 880h offset + (4d × i), where i=0d to 95d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MI																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- x = Undefined at reset.

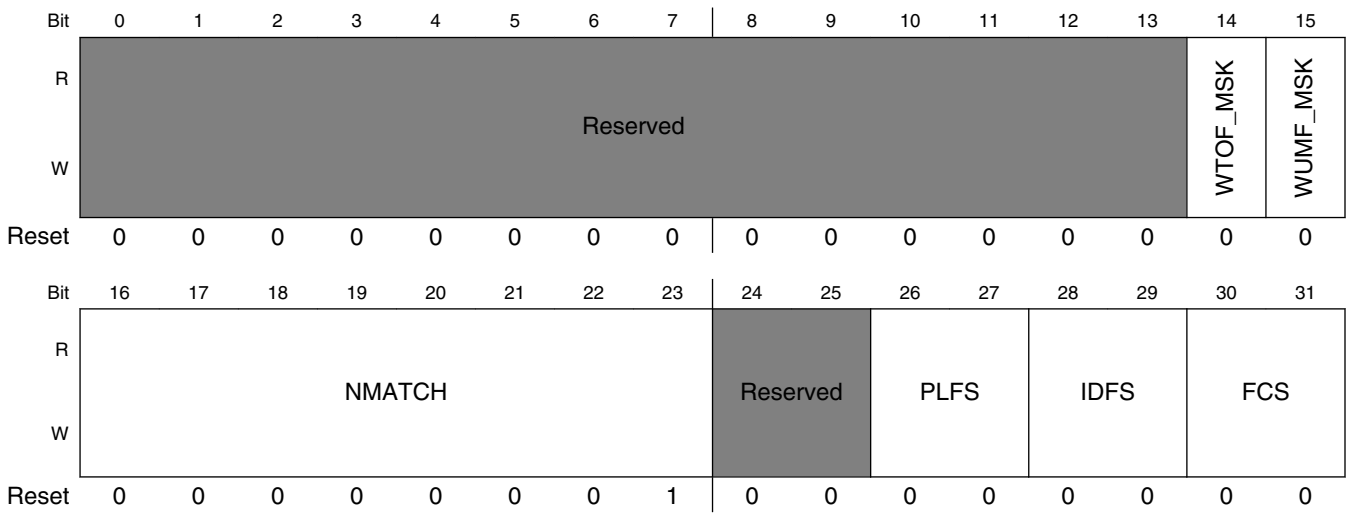
CAN_RXIMR*n* field descriptions

Field	Description
0–31 MI	Individual Mask Bits Each Individual Mask Bit masks the corresponding bit in both the Mailbox filter and Rx FIFO ID Filter Table element in distinct ways. For Mailbox filters, see the RXMGMASK register description. For Rx FIFO ID Filter Table elements, see the RXFGMASK register description. 0 The corresponding bit in the filter is "don't care." 1 The corresponding bit in the filter is checked.

44.4.23 Pretended Networking Control 1 Register (CAN_CTRL1_PN)

This register contains control bits for Pretended Networking mode filtering selection. Configure this register with the filter criteria to be used to receive wake up messages. It can be written in Freeze mode only, except WTOF_MSK and WUMF_MSK bits.

Address: 0h base + B00h offset = B00h



CAN_CTRL1_PN field descriptions

Field	Description
0–13 Reserved	This field is reserved.
14 WTOF_MSK	Wake Up by Timeout Flag Mask Bit This bit masks the generation of a wake up event originated by a timeout. 0 Timeout wake up event is disabled 1 Timeout wake up event is enabled
15 WUMF_MSK	Wake Up by Match Flag Mask Bit This bit masks the generation of a wake up event originated by a successful filtered Rx message. 0 Wake up match event is disabled 1 Wake up match event is enabled
16–23 NMATCH	Number of Messages Matching the Same Filtering Criteria This 8-bit field defines the number of times a given message must match the predefined filtering criteria for ID and/or PL before generating a wake up event. This quantity can be configured in the 1 to 255 range by using values from 0x01 to 0xFF, respectively. 00000001 Received message must match the predefined filtering criteria for ID and/or PL once before generating a wake up event. 00000010 Received message must match the predefined filtering criteria for ID and/or PL twice before generating a wake up event.

Table continues on the next page...

CAN_CTRL1_PN field descriptions (continued)

Field	Description
	... ... 11111111 Received message must match the predefined filtering criteria for ID and/or PL 255 times before generating a wake up event.
24–25 Reserved	This field is reserved.
26–27 PLFS	Payload Filtering Selection This 2-bit field selects the level of payload filtering to be applied when FlexCAN is under Pretended Networking mode. Filtering does not accept remote messages (RTR=1) when payload filtering is active. 00 Match upon a payload contents against an exact target value 01 Match upon a payload value greater than or equal to a specified target value 10 Match upon a payload value smaller than or equal to a specified target value 11 Match upon a payload value inside a range, greater than or equal to a specified lower limit and smaller than or equal a specified upper limit
28–29 IDFS	ID Filtering Selection This 2-bit field selects the level of ID filtering to be applied when FlexCAN is under Pretended Networking mode. In ID filtering, the IDE and RTR bits are also considered as part of reception filter if IDE_MSK and RTR_MSK bits in the CAN_FLT_ID2_IDMASK register are set. 00 Match upon a ID contents against an exact target value 01 Match upon a ID value greater than or equal to a specified target value 10 Match upon a ID value smaller than or equal to a specified target value 11 Match upon a ID value inside a range, greater than or equal to a specified lower limit and smaller than or equal a specified upper limit
30–31 FCS	Filtering Combination Selection This 2-bit field selects the filtering criteria to be applied when FlexCAN is under Pretended Networking mode. See Receive Process under Pretended Networking Mode for more details. 00 Message ID filtering only 01 Message ID filtering and payload filtering 10 Message ID filtering occurring a specified number of times. 11 Message ID filtering and payload filtering a specified number of times

44.4.24 Pretended Networking Control 2 Register (CAN_CTRL2_PN)

This register contains configuration bits for the timeout value under Pretended Networking mode. It can be written in Freeze mode only.

Address: 0h base + B04h offset = B04h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved																MATCHTO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_CTRL2_PN field descriptions

Field	Description
0–15 Reserved	This field is reserved.
16–31 MATCHTO	Timeout for No Message Matching the Filtering Criteria This 16-bit field defines a timeout value that generates a wake up event if CAN_MCR[PNET_EN] is asserted. If the timeout counter reaches the target value, when FlexCAN is under Pretended Networking mode, then a wake up event is generated. The timeout limit can be configured from 1 to 65535 to control an internal 16-bit up-count timer to produce a trigger upon reaching this configured value. The internal timer is incremented based on periodic time ticks, which period is 64 times the CAN Bit Time unit. When MATCHTO is configured with 0x0000 the timeout is disabled.

44.4.25 Pretended Networking Wake Up Match Register (CAN_WU_MTC)

This read-only register contains wake up information related to the matching processes performed while FlexCAN receives frames under Pretended Networking mode.

Address: 0h base + B08h offset = B08h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved														WTOF	WUMF
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MCOUNTER								Reserved							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_WU_MTC field descriptions

Field	Description
0–13 Reserved	This field is reserved.
14 WTOF	Wake Up by Timeout Flag Bit

Table continues on the next page...

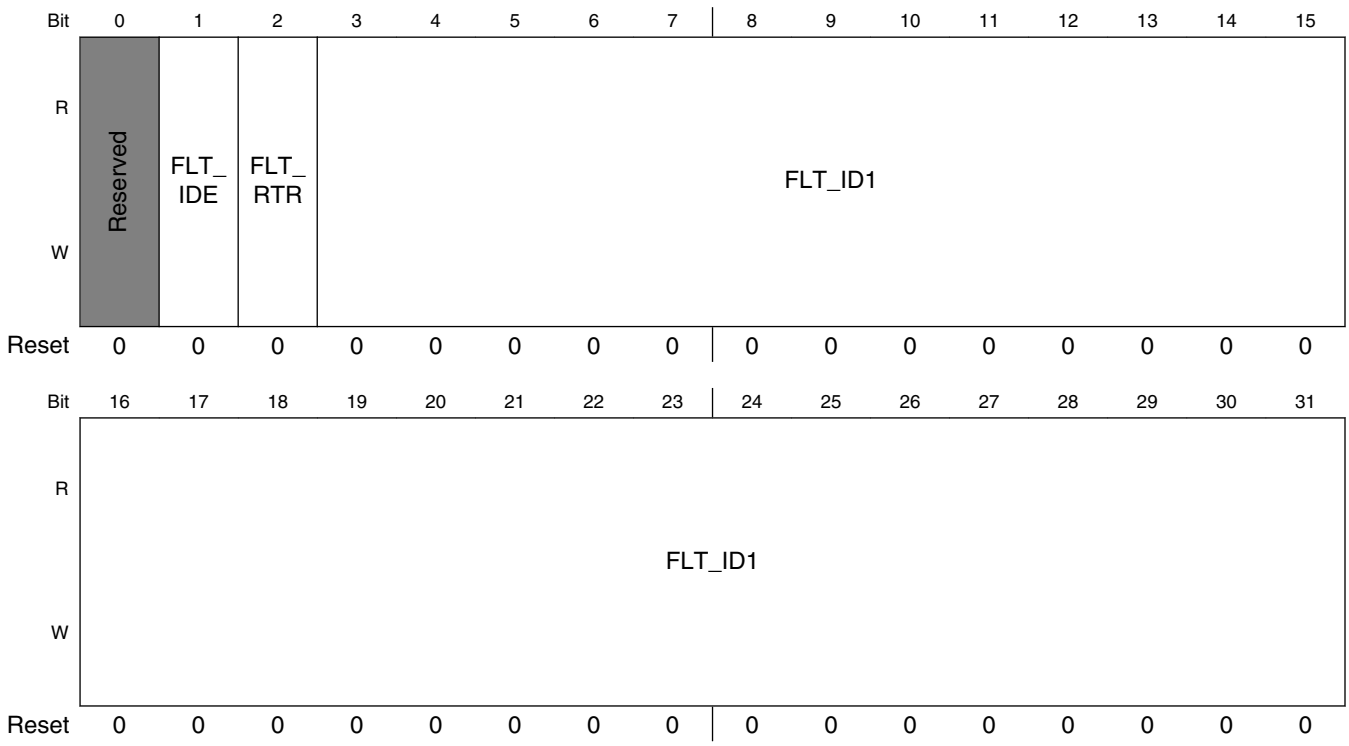
CAN_WU_MTC field descriptions (continued)

Field	Description
	This bit identifies whether the FlexCAN has detected a timeout event during a time interval defined by CAN_CTRL2_PN[MATCHTO]. This flag generates a wake up event if CAN_CTRL1_PN[WTOF_MSK] is enabled. 0 No wake up by timeout event detected 1 Wake up by timeout event detected
15 WUMF	Wake Up by Match Flag Bit This bit identifies whether the FlexCAN has detected a matching Rx incoming message, which passed the filtering criteria specified in CAN_CTRL1_PN register. This flag generates a wake up event if CAN_CTRL1_PN[WUMF_MSK] is enabled. 0 No wake up by match event detected 1 Wake up by match event detected
16–23 MCOUNTER	Number of Matches while in Pretended Networking This 8-bit field reports the number of times a given message have been matched the predefined filtering criteria for ID and/or PL before a wake up event. This register is reset by FlexCAN when it enters in Pretended Networing mode, and is not affected by soft reset.
24–31 Reserved	This field is reserved.

44.4.26 Pretended Networking ID Filter 1 Register (CAN_FLT_ID1)

This register contains FLT_ID1 target value, as well as, IDE and RTR target values used to filter incoming message ID. The FLT_ID1 is used either for equal to, smaller than, greater than comparisons, or as the lower limit value in a ID range detection. It can be written in Freeze mode only.

Address: 0h base + B0Ch offset = B0Ch



CAN_FLT_ID1 field descriptions

Field	Description
0 Reserved	This field is reserved.
1 FLT_IDE	ID Extended Filter This bit identifies whether the frame format is standard or extended. It is used as part of the ID reception filter. 0 Accept standard frame format 1 Accept extended frame format
2 FLT_RTR	Remote Transmission Request Filter This bit identifies whether the frame is remote or not. It is used as part of the ID reception filter. 0 Reject remote frame (accept data frame) 1 Accept remote frame

Table continues on the next page...

CAN_FLT_ID1 field descriptions (continued)

Field	Description
3–31 FLT_ID1	ID Filter 1 for Pretended Networking filtering This 29-bit field defines either the 29 bits of a extended frame format, considering all bits, or the 11 bits of a standard frame format, considering just the 11 leftmost bits.

44.4.27 Pretended Networking DLC Filter Register (CAN_FLT_DLC)

This register contains the DLC inside range target values (FLT_DLC_LO and FLT_DLC_HI) used to filter incoming message. The DLC range is used only for payload filtering. It can be written in Freeze mode only.

NOTE

When a fixed quantity of data bytes is required, both FLT_DLC_LO and FLT_DLC_HI need to be configured with the same value, otherwise a range of DLC is considered for filtering (see Receive Process under Pretended Networking Mode).

Address: 0h base + B10h offset = B10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved												FLT_DLC_LO				Reserved												FLT_DLC_HI			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

CAN_FLT_DLC field descriptions

Field	Description
0–11 Reserved	This field is reserved.
12–15 FLT_DLC_LO	Lower Limit for Length of Data Bytes Filter This field specifies the lower limit for the number of data bytes considered valid for payload comparison. It is used as part of payload reception filter.
16–27 Reserved	This field is reserved.
28–31 FLT_DLC_HI	Upper Limit for Length of Data Bytes Filter This field specifies the upper limit for the number of data bytes considered valid for payload comparison. It is used as part of payload reception filter.

44.4.28 Pretended Networking Payload Low Filter 1 Register (CAN_PL1_LO)

This register contains Payload Filter 1 low order bits of the target value used to filter incoming message payload. It is used either for “equal to”, “smaller than or equal”, “greater than or equal” comparisons, or as the lower limit value in a payload range detection. It can be written in Freeze mode only.

Address: 0h base + B14h offset = B14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Data_byte_0								Data_byte_1								Data_byte_2								Data_byte_3							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_PL1_LO field descriptions

Field	Description
0–7 Data_byte_0	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to the data byte 0.
8–15 Data_byte_1	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to the data byte 1.
16–23 Data_byte_2	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to the data byte 2.
24–31 Data_byte_3	Payload Filter 1 low order bits for Pretended Networking payload filtering corresponding to the data byte 3.

44.4.29 Pretended Networking Payload High Filter 1 Register (CAN_PL1_HI)

This register contains Payload Filter 1 high order bits of the target value used to filter incoming message payload. It is used either for “equal to”, “smaller than or equal to”, “greater than or equal to” comparisons, or as the lower limit value in a payload range detection. It can be written in Freeze mode only.

Address: 0h base + B18h offset = B18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Data_byte_4								Data_byte_5								Data_byte_6								Data_byte_7							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_PL1_HI field descriptions

Field	Description
0–7 Data_byte_4	Payload Filter 1 high order bits for Pretended Networking payload filtering corresponding to the data byte 4.
8–15 Data_byte_5	Payload Filter 1 high order bits for Pretended Networking payload filtering corresponding to the data byte 5.
16–23 Data_byte_6	Payload Filter 1 high order bits for Pretended Networking payload filtering corresponding to the data byte 6.
24–31 Data_byte_7	Payload Filter 1 high order bits for Pretended Networking payload filtering corresponding to the data byte 7.

44.4.30 Pretended Networking ID Filter 2 Register / ID Mask Register (CAN_FLT_ID2_IDMASK)

This register contains FLT_ID2 target value used only as the upper limit value in ID range detection. Also, when exact ID filtering criteria is selected, this register is used to store the ID mask. IDE_MSK and RTR_MSK bits are used in both ID filtering (exact and range) to enable FLT_IDE and FLT_RTR respectively to be used as part of ID reception filter. This register can be written in Freeze mode only.

Address: 0h base + B1Ch offset = B1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved	IDE_MSK	RTR_MSK	FLT_ID2_IDMASK												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	FLT_ID2_IDMASK															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_FLT_ID2_IDMASK field descriptions

Field	Description
0 Reserved	This field is reserved.
1 IDE_MSK	ID Extended Mask Bit This bit indicates whether the frame format (standard / extended) is used as part of the ID reception filter. 0 The corresponding bit in the filter is “don’t care” 1 The corresponding bit in the filter is checked
2 RTR_MSK	Remote Transmission Request Mask Bit This bit indicates whether the frame type (data / remote) is part of the ID reception filter. 0 The corresponding bit in the filter is “don’t care” 1 The corresponding bit in the filter is checked
3–31 FLT_ID2_ IDMASK	ID Filter 2 for Pretended Networking Filtering / ID Mask Bits for Pretended Networking ID Filtering This register is used to define either the ID filter value in extended frame format (29 bits), considering the FLT_ID2[28:0], or the ID filter value in standard frame format (11 bits), considering the FLT_ID2[28:18] (other bits in the [17:0] range have no meaning). Used only in range of ID filtering. It can also be used to define either the mask values for the extended frame format (29 bits), considering the IDMASK[28:0], or for the standard frame format (11 bits), considering the IDMASK[28:18] (other bits in the [17:0] range have no meaning). Used only in exact ID filtering.

44.4.31 Pretended Networking Payload Low Filter 2 Register / Payload Low Mask Register (CAN_PL2_PLMASK_LO)

This register has two functions. First, it contains the low order bits for the Payload Filter 2 used only as the upper limit value in a payload range detection. Second, when exact payload filtering criteria is selected, this register is used as payload mask. Otherwise, this register is unused. It can be written in Freeze mode only.

Address: 0h base + B20h offset = B20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Data_byte_0								Data_byte_1								Data_byte_2								Data_byte_3							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_PL2_PLMASK_LO field descriptions

Field	Description
0–7 Data_byte_0	Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to the data byte 0.
8–15 Data_byte_1	Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to the data byte 1.
16–23 Data_byte_2	Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to the data byte 2.

Table continues on the next page...

CAN_PL2_PLMASK_LO field descriptions (continued)

Field	Description
24–31 Data_byte_3	Payload Filter 2 low order bits / Payload Mask low order bits for Pretended Networking payload filtering corresponding to the data byte 3.

44.4.32 Pretended Networking Payload High Filter 2 low order bits / Payload High Mask Register (CAN_PL2_PLMASK_HI)

This register has two functions. First, it contains the high order bits for the Payload Filter 2 used only as the upper limit value in a payload range detection. Second, when exact payload filtering criteria is selected, this register is used as payload mask. Otherwise, this register is unused. It can be written in Freeze mode only.

Address: 0h base + B24h offset = B24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Data_byte_4								Data_byte_5								Data_byte_6								Data_byte_7							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_PL2_PLMASK_HI field descriptions

Field	Description
0–7 Data_byte_4	Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to the data byte 4.
8–15 Data_byte_5	Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to the data byte 5.
16–23 Data_byte_6	Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to the data byte 6.
24–31 Data_byte_7	Payload Filter 2 high order bits / Payload Mask high order bits for Pretended Networking payload filtering corresponding to the data byte 7.

44.4.33 Wake Up Message Buffer Register for C/S (CAN_WMB*n*_CS)

Each of the four WMBs contains a register to store the Control Status (C/S) information (IDE, RTR and DLC fields) of an incoming Rx message.

NOTE

The C/S registers are located at 0xB40 for WMB0, 0xB50 for WMB1, 0xB60 for WMB2, and 0xB70 for WMB3.

Address: 0h base + B40h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved									SRR	IDE	RTR	DLC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_WMB*n*_CS field descriptions

Field	Description
0–8 Reserved	This field is reserved.
9 SRR	Substitute Remote Request This bit can be received either recessive or dominant.
10 IDE	ID Extended Bit This bit identifies whether the frame format is standard or extended 0 Frame format is standard 1 Frame format is extended
11 RTR	Remote Transmission Request Bit This bit identifies whether the frame is remote or not. 0 Frame is data one (not remote) 1 Frame is a remote one
12–15 DLC	Length of Data in Bytes This 4-bit field is the length (in bytes) of the Rx data received when FlexCAN is in Pretended Networking mode. This 4-bit field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. The DLC field indicates which data bytes are valid.
16–31 Reserved	This field is reserved.

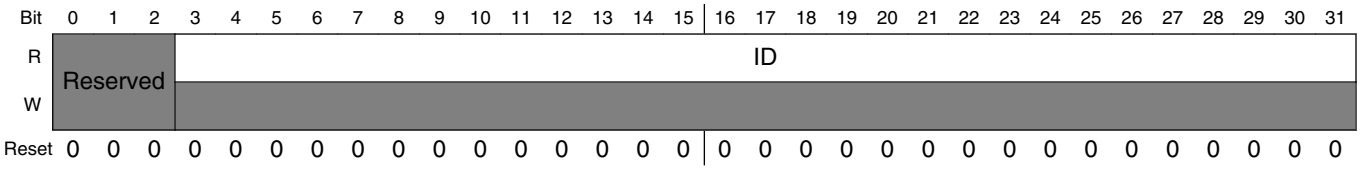
44.4.34 Wake Up Message Buffer Register for ID (CAN_WMB*n*_ID)

Each of the four WMBs contains a register to store the ID information of an incoming Rx message.

NOTE

The ID registers are located at 0xB44 for WMB0, 0xB54 for WMB1, 0xB64 for WMB2, and 0xB74 for WMB3.

Address: 0h base + B44h offset + (16d × i), where i=0d to 3d



CAN_WMB*n*_ID field descriptions

Field	Description
0–2 Reserved	This field is reserved.
3–31 ID	Received ID under Pretended Networking mode This register stores either the 29 bits of the extended frame format (considering the ID[0:28] field), or the 11 bits of the standard frame format (considering the ID[0:10] field only, the remaining bits in the ID[11:28] range have no meaning).

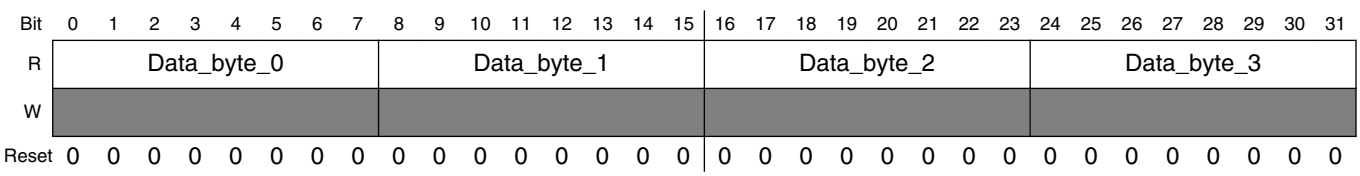
44.4.35 Wake Up Message Buffer Register for Data 0-3 (CAN_WMB*n*_D03)

Each of the four WMBs contains a register to store the data bytes 0 to 3 of the payload information of an incoming Rx message. This register's content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.

NOTE

The Data 0-3 registers are located at 0xB48 for WMB0, 0xB58 for WMB1, 0xB68 for WMB2, and 0xB78 for WMB3.

Address: 0h base + B48h offset + (16d × i), where i=0d to 3d



CAN_WMB*n*_D03 field descriptions

Field	Description
0–7 Data_byte_0	Received payload corresponding to the data byte 0 under Pretended Networking mode
8–15 Data_byte_1	Received payload corresponding to the data byte 1 under Pretended Networking mode
16–23 Data_byte_2	Received payload corresponding to the data byte 2 under Pretended Networking mode
24–31 Data_byte_3	Received payload corresponding to the data byte 3 under Pretended Networking mode

44.4.36 Wake Up Message Buffer Register Data 4-7 (CAN_WMB*n*_D47)

Each of the four WMBs contains a register to store the data bytes 4 to 7 of the payload information of an incoming Rx message. This register's content is cleared when the incoming matched message is either a remote frame (RTR=1) or a data frame with DLC=0.

NOTE

The Data 4-7 registers are located at 0xB4C for WMB0, 0xB5C for WMB1, 0xB6C for WMB2, and 0xB7C for WMB3.

Address: 0h base + B4Ch offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Data_byte_4								Data_byte_5								Data_byte_6								Data_byte_7							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CAN_WMB*n*_D47 field descriptions

Field	Description
0–7 Data_byte_4	Received payload corresponding to the data byte 4 under Pretended Networking mode
8–15 Data_byte_5	Received payload corresponding to the data byte 5 under Pretended Networking mode
16–23 Data_byte_6	Received payload corresponding to the data byte 6 under Pretended Networking mode
24–31 Data_byte_7	Received payload corresponding to the data byte 7 under Pretended Networking mode

44.4.37 CAN FD Control Register (CAN_FDCTRL)

This register contains control bits for the CAN FD operation. It also defines the data size of Message Buffers allocated in different partitions of RAM (memory blocks) as described in the table below.

When 8 bytes payload is selected:

- Block R0 allocates MB0 to MB31.
- Block R1 allocates MB32 to MB63.
- Block R2 allocates MB64 to MB95.

When more than 8 bytes payload is selected, the maximum number of MBs in a block is limited as described below:

Table 44-5. Number of Message Buffers

Payload Size	Maximum number of Message Buffers per RAM block
8 bytes	32
16 bytes	21
32 bytes	12
64 bytes	7

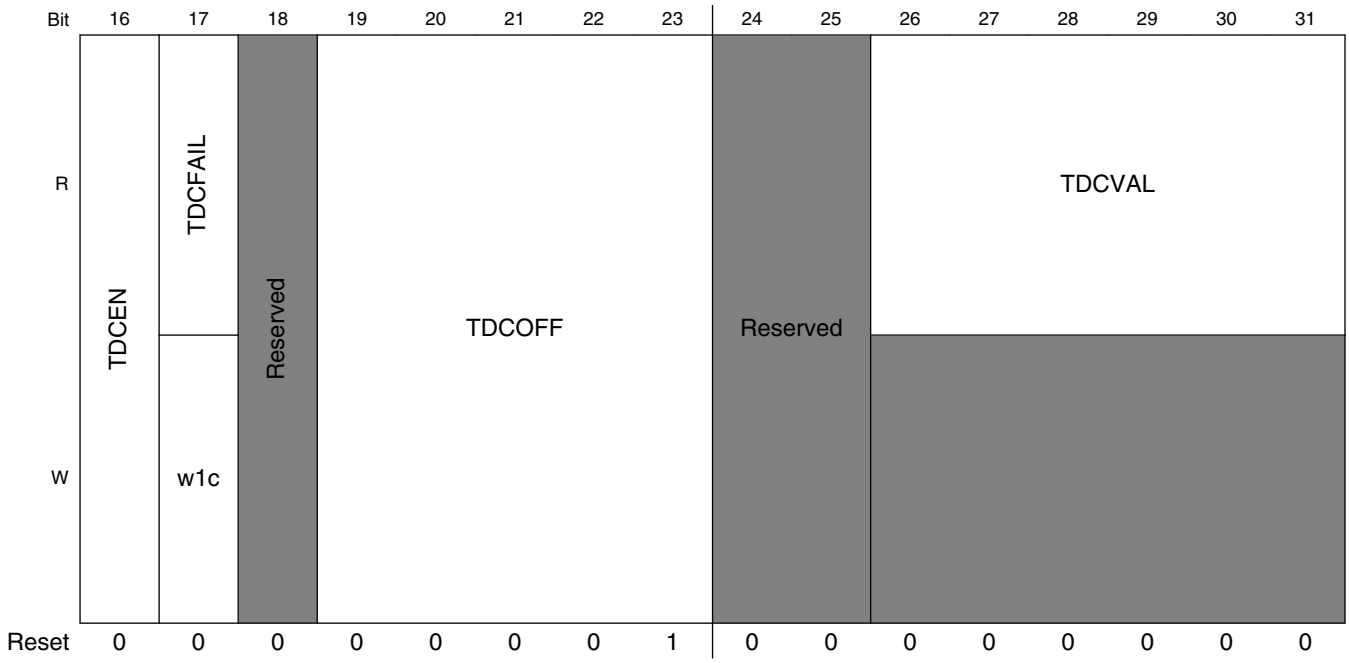
NOTE

One memory block fits exactly 32 MBs with 8 bytes payload. For the other options of payload sizes, empty memory may exist between last MB in a block and the beginning of the next block. This empty memory corresponds to less than one MB, and must not be used.

The contents of this register are not affected by soft reset.

Address: 0h base + C00h offset = C00h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	FDRATE	Reserved				Reserved		Reserved	MBDSR2		Reserved	MBDSR1		Reserved	MBDSR0	
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



CAN_FDCTRL field descriptions

Field	Description
0 FDRATE	Bit Rate Switch Enable This bit enables the effect of the Bit Rate Switch (BRS bit) during the data phase of Tx messages. The CPU can write this bit any time. However, its effect turns active only when the CAN bus is in Wait for Bus Idle, Bus Idle or Bus Off state, or when the current frame under reception or transmission reaches the interframe space. By negating the CAN_FDCTRL[FDRATE] bit, the CPU can force all bits in CAN FD messages to be transmitted in nominal bit rate, despite of the value in the BRS bit of the Tx MBs. 0 Transmit a frame in nominal rate. The BRS bit in the Tx MB has no effect. 1 Transmit a frame with bit rate switching if the BRS bit in the Tx MB is recessive.
1–4 Reserved	This field is reserved.
5–6 Reserved	This field is reserved.
7 Reserved	This field is reserved.
8–9 MBDSR2	Message Buffer Data Size for Region 2 This two bit field selects the data size (8, 16, 32 or 64 bytes) for the region R2 of Message Buffers allocated in RAM. It can be written in Freeze Mode only. 00 Selects 8 bytes per Message Buffer. 01 Selects 16 bytes per Message Buffer. 10 Selects 32 bytes per Message Buffer. 11 Selects 64 bytes per Message Buffer.

Table continues on the next page...

CAN_FDCTRL field descriptions (continued)

Field	Description
10 Reserved	This field is reserved.
11–12 MBDSR1	Message Buffer Data Size for Region 1 This two bit field selects the data size (8, 16, 32 or 64 bytes) for the region R1 of Message Buffers allocated in RAM. It can be written in Freeze Mode only. 00 Selects 8 bytes per Message Buffer. 01 Selects 16 bytes per Message Buffer. 10 Selects 32 bytes per Message Buffer. 11 Selects 64 bytes per Message Buffer.
13 Reserved	This field is reserved.
14–15 MBDSR0	Message Buffer Data Size for Region 0 This two bit field selects the data size (8, 16, 32 or 64 bytes) for the region R0 of Message Buffers allocated in RAM. It can be written in Freeze Mode only. 00 Selects 8 bytes per Message Buffer. 01 Selects 16 bytes per Message Buffer. 10 Selects 32 bytes per Message Buffer. 11 Selects 64 bytes per Message Buffer.
16 TDCEN	Transceiver Delay Compensation Enable This bit can be used to enable and disable the TDC feature. It can be written in Freeze mode only. **NOTE:** TDC must be disabled when the Loop Back Mode is enabled (see CAN_CTRL1[LPB] register). 0 TDC is disabled 1 TDC is enabled
17 TDCFAIL	Transceiver Delay Compensation Fail This bit indicates when the Transceiver Delay Compensation (TDC) mechanism is out of range, unable to compensate the transceiver's loop delay and successfully compare the delayed received bits to the transmitted ones (see Transceiver Delay Compensation. TDCFAIL sets in the first time FlexCAN detects the out of range condition. The CPU needs to write 1 to clear it. 0 Measured loop delay is in range. 1 Measured loop delay is out of range.
18 Reserved	This field is reserved.
19–23 TDCOFF	Transceiver Delay Compensation Offset This bit field contains the offset value to be added to the measured transceiver's loop delay in order to define the position of the delayed comparison point when bit rate switching is active. See Transceiver Delay Compensation for more details on how the loop delay measurement is performed. TDCOFF can be written in Freeze mode only. Its value can be defined in Protocol Engine (PE) Clock periods (CANCLK, see Protocol timing for more details), and must be selected to be smaller than the CAN bit duration in the data bit rate for proper operation.

Table continues on the next page...

CAN_FDCTRL field descriptions (continued)

Field	Description
	NOTE: It is not recommended to use TDCOFF equal to zero.
24–25 Reserved	This field is reserved.
26–31 TDCVAL	Transceiver Delay Compensation Value This register contains the value of the transceiver loop delay measured from the transmitted EDL to R0 transition edge to the respective received one added to the TDCOFF value specified in the CAN_FDCTRL register. This value is an integer multiple of the Protocol Engine (PE) Clock period (CANCLK). See Protocol timing for more details on how the loop delay measurement is performed.

44.4.38 CAN FD Bit Timing Register (CAN_FDCBT)

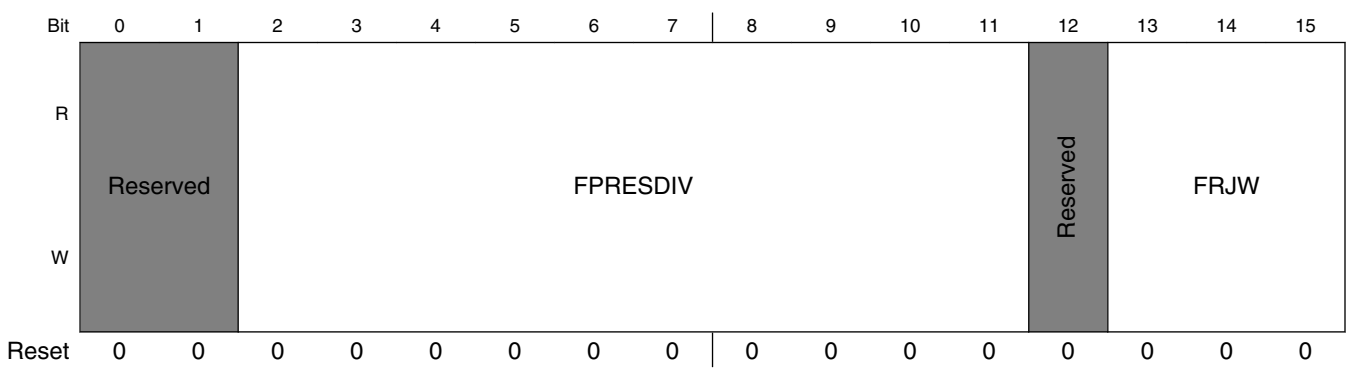
This register stores the CAN bit timing variables used in the data phase of CAN FD messages when the CAN_FDCTRL[FDRATE] is set, compatible with CAN FD specification. FPRESDIV, FPROPSEG, FPSEG1, FPSEG2 and FRJW are used to define the time quantum duration, the number of time quanta per CAN bit and the sample point position for the data bit rate portion of a CAN FD message with the BRS bit set.

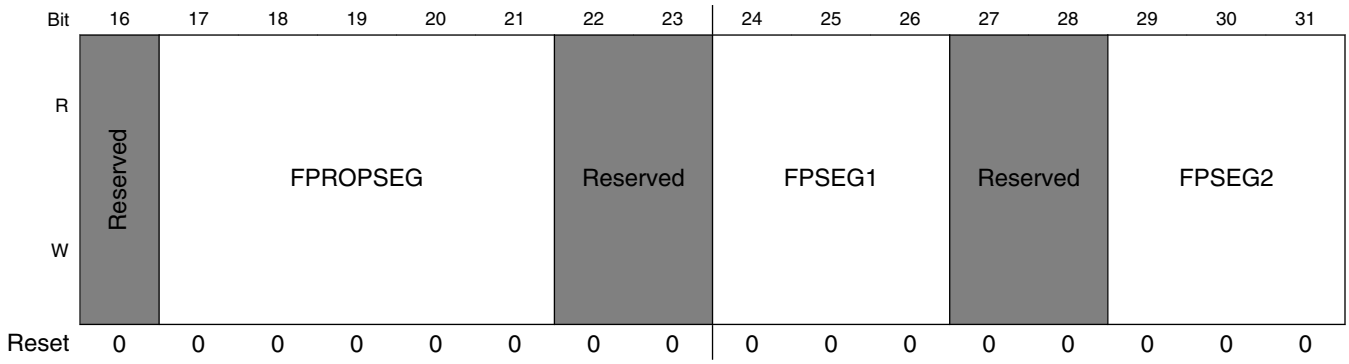
The contents of this register are not affected by soft reset.

NOTE

The sum of the Fast Propagation Segment (FPROPSEG) and Fast Phase Segment 1 (FPSEG1) must be at least two time quanta.

Address: 0h base + C04h offset = C04h





CAN_FDCBT field descriptions

Field	Description
0–1 Reserved	This field is reserved.
2–11 FPRESDIV	Fast Prescaler Division Factor This 10-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency in the data bit rate portion of a CAN FD message with the BRS bit set. The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (FPRESDIV + 1). **NOTE:** To minimize errors when processing FD frames, use the same value for FPRESDIV and PRESDIV (in CAN_CBT or CAN_CTRL1). For more details refer to the first NOTE in section CAN FD frames.
12 Reserved	This field is reserved.
13–15 FRJW	Fast Resync Jump Width This 3-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization in the data bit rate portion of a CAN FD message with the BRS bit set. One time quantum is equal to the Sclock period. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = FSJW + 1.
16 Reserved	This field is reserved.
17–21 FPROPSEG	Fast Propagation Segment This 5-bit field defines the length of the Propagation Segment in the bit time in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = FPROPSEG × Time-Quanta. Time-Quantum = one Sclock period.
22–23 Reserved	This field is reserved.
24–26 FPSEG1	Fast Phase Segment 1

Table continues on the next page...

CAN_FDCBT field descriptions (continued)

Field	Description
	This 3-bit field defines the length of Phase Segment 1 in the bit time in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Segment 1 = (FPSEG1 + 1) × Time-Quanta. Time-Quantum = one Sclock period.
27–28 Reserved	This field is reserved.
29–31 FPSEG2	Fast Phase Segment 2 This 3-bit field defines the length of Phase Segment 2 in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Segment 2 = (FPSEG2 + 1) × Time-Quanta. Time-Quantum = one Sclock period.

44.4.39 CAN FD CRC Register (CAN_FDCRC)

This register provides information about the CRC of transmitted messages.

FlexCAN uses different CRC polynomials for different frame formats, as shown below.

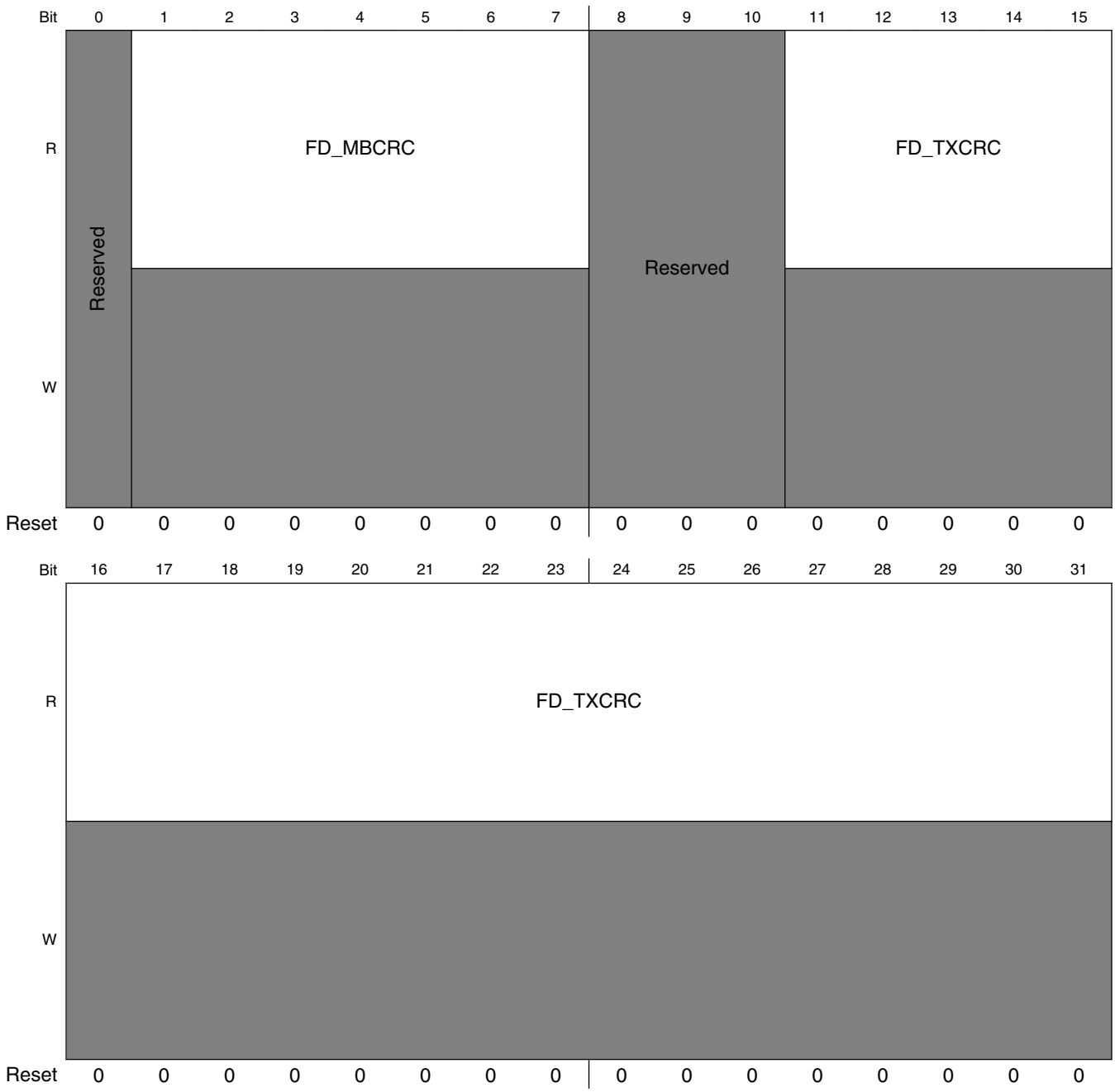
The CRC_15 polynomial is used for all frames in CAN format. The CRC_17 polynomial is used for frames in CAN FD format with a DATA FIELD up to sixteen bytes. The CRC_21 polynomial is used for frames in CAN FD format with a DATA FIELD longer than sixteen bytes. Each polynomial shown below results in a Hamming Distance of 6. This register is updated at the same time the Tx Interrupt Flag is asserted.

CRC_15 = 0xC599: $(x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1)$

CRC_17 = 0x3685B: $(x^{17} + x^{16} + x^{14} + x^{13} + x^{11} + x^6 + x^4 + x^3 + x^1 + 1)$

CRC_21 = 0x302899: $(x^{21} + x^{20} + x^{13} + x^{11} + x^7 + x^4 + x^3 + 1)$

Address: 0h base + C08h offset = C08h



CAN_FDCRC field descriptions

Field	Description
0 Reserved	This field is reserved.
1–7 FD_MBCRC	CRC Mailbox Number for FD_TXCRC This field indicates the number of the Mailbox corresponding to the value in FD_TXCRC field, for both FD and non-FD frames. It reports the same information as in MBCRC bit field in CAN_CRCR register.

Table continues on the next page...

CAN_FDCRC field descriptions (continued)

Field	Description
8–10 Reserved	This field is reserved.
11–31 FD_TXCRC	Extended Transmitted CRC value This 21-bit field contains the CRC value calculated over the most recent transmitted message. Different CRC polynomials are used for different frame formats. A 15-bit polynomial, CRC_15, is used for all frames in CAN format. The second 17-bit polynomial, CRC_17, is used for frames in CAN FD format with a data field up to sixteen bytes long. The third 21-bit polynomial, CRC_21, is used for frames in CAN FD format with a data field longer than sixteen bytes. For CRC_15 and CRC_17, the 6 most significant bits and the 4 most significant bits are reported as zeros, respectively. For CRC_15, this register has the same content as CRC Register.

44.4.40 Message buffer structure

The message buffer structure used by the FlexCAN module is represented in the following figure. Both Extended (29-bit identifier) and Standard (11-bit identifier) frames used in the CAN specification (Version 2.0 Part B) are represented. Each individual MB is formed by 16, 24, 40 or 72 bytes, depending on the quantity of data bytes allocated for the message payload: 8, 16, 32 or 64 data bytes, respectively.

The memory area from 0x80 to 0x67F is used by the mailboxes. When CAN FD is enabled, the exact address for each MB depends on the size of its payload. See FlexCAN Memory Partition for CAN FD for more detailed information.

Table 44-6. Message buffer structure - example with 64 bytes payload

	0	1	2	3	4 7	8	9	10	11	12 13 14 15	16 23	24 31
0x0	EDL	BRS	ESI		CODE		SRR	IDE	RTR	DLC	TIME STAMP	
0x4	PRIO			ID (Standard/Extended)							ID (Extended)	
0x8	Data Byte 0					Data Byte 1					Data Byte 2	Data Byte 3
0xC	Data Byte 4					Data Byte 5					Data Byte 6	Data Byte 7
0x10	Data Byte 8					Data Byte 9					Data Byte 10	Data Byte 11
0x14	Data Byte 12					Data Byte 13					Data Byte 14	Data Byte 15
0x18	Data Byte 16					Data Byte 17					Data Byte 18	Data Byte 19
0x1C	Data Byte 20					Data Byte 21					Data Byte 22	Data Byte 23
0x20	Data Byte 24					Data Byte 25					Data Byte 26	Data Byte 27
0x24	Data Byte 28					Data Byte 29					Data Byte 30	Data Byte 31
0x28	Data Byte 32					Data Byte 33					Data Byte 34	Data Byte 35
0x2C	Data Byte 36					Data Byte 37					Data Byte 38	Data Byte 39

Table continues on the next page...

Table 44-6. Message buffer structure - example with 64 bytes payload (continued)

0x30	Data Byte 40	Data Byte 41	Data Byte 42	Data Byte 43
0x34	Data Byte 44	Data Byte 45	Data Byte 46	Data Byte 47
0x38	Data Byte 48	Data Byte 49	Data Byte 50	Data Byte 51
0x3C	Data Byte 52	Data Byte 53	Data Byte 54	Data Byte 55
0x40	Data Byte 56	Data Byte 57	Data Byte 58	Data Byte 59
0x44	Data Byte 60	Data Byte 61	Data Byte 62	Data Byte 63

= Unimplemented or Reserved

EDL - Extended Data Length

This bit distinguishes between CAN format and CAN FD format frames. The EDL bit must not be set for Message Buffers configured to RANSWER with code field 0b1010 (see Table below).

BRS - Bit Rate Switch

This bit defines whether the bit rate is switched inside a CAN FD format frame.

ESI - Error State Indicator

This bit indicates if the transmitting node is error active or error passive.

CODE - Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the FlexCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 44-7 and Table 44-8. See Functional description for additional information.

Table 44-7. Message buffer code for Rx buffers

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0b0000: INACTIVE - MB is not active.	INACTIVE	-	-	-	MB does not participate in the matching process.
0b0100: EMPTY - MB is active and empty.	EMPTY	-	FULL	-	When a frame is received successfully (after the Move-in) process), the CODE field is automatically updated to FULL.

Table continues on the next page...

Table 44-7. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0b0010: FULL - MB is full.	FULL	Yes	FULL	-	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code still remains FULL. See Matching process for matching details related to FULL code.
		No	OVERRUN	-	If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. See Matching process for details about overrun behavior.
0b0110: OVERRUN - MB is being overwritten into a full buffer.	OVERRUN	Yes	FULL	-	If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB then the code returns to FULL.
		No	OVERRUN	-	If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. See Matching process for details about overrun behavior.

Table continues on the next page...

Table 44-7. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0b1010: RANSWER[4] - A frame was configured to recognize a Remote Request Frame and transmit a Response Frame in return.[5]	RANSWER	-	TANSWER(0b1110)	0	A Remote Answer was configured to recognize a remote request frame received. After that an MB is set to transmit a response frame. The code is automatically changed to TANSWER (0b1110). See Matching process for details. If CAN_CTRL2[RRS] is negated, transmit a response frame whenever a remote request frame with the same ID is received.
		-	-	1	This code is ignored during matching and arbitration process. See Matching process for details.
CODE[0]=1: BUSY - FlexCAN is updating the contents of the MB. The CPU must not access the MB.	BUSY[6]	-	FULL	-	Indicates that the MB is being updated. It will be negated automatically and does not interfere with the next CODE.
		-	OVERRUN	-	

1. SRV: Serviced MB. MB was read and unlocked by reading TIMER or other MB.
2. A frame is considered a successful reception after the frame to be moved to MB (move-in process). See Move-in for details.
3. Remote Request Stored bit, see "Control 2 Register (CAN_CTRL2)" for details.
4. Code 0b1010 is not considered Tx and an MB with this code should not be aborted.
5. Code 0b1010 must be used in Message Buffers configured in CAN FD format, having the EDL bit set.
6. Note that for Tx MBs, the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 44-8. Message buffer code for Tx buffers

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
0b1000: INACTIVE - MB is not active	INACTIVE	-	-	MB does not participate in arbitration process.
0b1001: ABORT - MB is aborted	ABORT	-	-	MB does not participate in arbitration process.
0b1100: DATA - MB is a Tx Data Frame (MB RTR must be 0)	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
0b1100: REMOTE - MB is a Tx Remote Request Frame (MB RTR must be 1)	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID.
0b1110: TANSWER - MB is a Tx Response Frame from an incoming Remote Request Frame	TANSWER	-	RANSWER	This is an intermediate code that is automatically written to the MB by the CHI as a result of a match to a remote request frame. The remote response frame will be transmitted unconditionally once, and then the code will automatically return to RANSWER (0b1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. See Matching process and Arbitration process for details.

SRR - Substitute Remote Request

Fixed recessive bit, used only in extended format. It must be set to one by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as an arbitration loss.

1 = Recessive value is compulsory for transmission in extended format frames

0 = Dominant is not a valid value for transmission in extended format frames

IDE - ID Extended Bit

This field identifies whether the frame format is standard or extended.

1 = Frame format is extended

0 = Frame format is standard

RTR - Remote Transmission Request

This bit affects the behavior of remote frames and is part of the reception filter. See Table 44-7, Table 44-8, and the description of the RRS bit in Control 2 Register (CAN_CTRL2) for additional details.

If FlexCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as an arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FlexCAN module treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission.

1 = Indicates the current MB may have a remote request frame to be transmitted if MB is Tx. If the MB is Rx then incoming remote request frames may be stored.

0 = Indicates the current MB has a data frame to be transmitted. In Rx MB it may be considered in matching processes.

NOTE

When configuring CAN FD frames, the RTR bit must be negated.

DLC - Length of Data in Bytes

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x8 through 0xF of the MB space (see Table 44-6) . In reception, this field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field (see Table 44-9).

TIME STAMP - Free-Running Counter Time Stamp

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.

PRIO - Local priority

This 3-bit field is used only when LPRIO_EN bit is set in CAN_MCR, and it only makes sense for Tx mailboxes. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration process.

ID - Frame Identifier

In standard frame format, only the 11 most significant bits (3 to 13) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification in both receive and transmit cases.

DATA BYTE 0 to 63 - Data Field

Up to sixty four bytes can be used for a data frame, depending on the size of payload selected for the Message Buffers.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE (*n*) is valid only if *n* is less than DLC as shown in the table below.

Table 44-9. DATA BYTEs validity

DLC	Valid DATA BYTEs
0	none
1	DATA BYTE 0
2	DATA BYTE 0 to 1
3	DATA BYTE 0 to 2
4	DATA BYTE 0 to 3
5	DATA BYTE 0 to 4
6	DATA BYTE 0 to 5
7	DATA BYTE 0 to 6
8	DATA BYTE 0 to 7
9	DATA BYTE 0 to 11
10	DATA BYTE 0 to 15
11	DATA BYTE 0 to 19
12	DATA BYTE 0 to 23
13	DATA BYTE 0 to 31
14	DATA BYTE 0 to 47
15	DATA BYTE 0 to 63

44.4.41 FlexCAN Memory Partition for CAN FD

When CAN FD is enabled, the FlexCAN RAM can be partitioned in blocks of 512 bytes. Each block can accommodate a number of Message Buffers which depends on the configuration provided by CAN_FDCTRL[MBDSRn] bit fields as shown in table below.

Table 44-10. RAM partition

RAM block	Number of MBs with 8 bytes (default range)	Size control bit field in CAN_FDCTRL register	Number of MBs of different sizes, per block
0	0 to 31	MBDSR0	MBDSR0=00, 32 MBs with 8 bytes payload MBDSR0=01, 21 MBs with 16 bytes payload MBDSR0=10, 12 MBs with 32 bytes payload MBDSR0=11, 7 MBs with 64 bytes payload
1	32 to 63	MBDSR1	MBDSR1=00, 32 MBs with 8 bytes payload MBDSR1=01, 21 MBs with 16 bytes payload MBDSR1=10, 12 MBs with 32 bytes payload MBDSR1=11, 7 MBs with 64 bytes payload
2	64 to 95	MBDSR2	MBDSR2=00, 32 MBs with 8 bytes payload MBDSR2=01, 21 MBs with 16 bytes payload MBDSR2=10, 12 MBs with 32 bytes payload MBDSR2=11, 7 MBs with 64 bytes payload

When payload sizes of 16, 32 or 64 bytes are configured in some or all RAM blocks, the total number of MBs and its respective number order may differ from the default configuration of 8 bytes. For example, suppose Block0 is configured to 8 bytes payload, Block1 to 16 bytes, Block2 to 32 bytes, than the following table indicates how the Message Buffers will be arranged into RAM.

Table 44-11. RAM partition example

RAM block	Payload size	Number of MBs in the RAM block	Message Buffer range
0	CAN_FDCTRL[MBDSR0]=00, 8 bytes payload	32	0 to 31

Table continues on the next page...

Table 44-11. RAM partition example (continued)

RAM block	Payload size	Number of MBs in the RAM block	Message Buffer range
1	CAN_FDCTRL[MBDSR1]=01, 16 bytes payload	21	32 to 52
2	CAN_FDCTRL[MBDSR2]=10, 32 bytes payload	12	53 to 64

44.4.42 FlexCAN message buffer memory map

The FlexCAN memory buffers are allocated in memory according to the tables below.

Table 44-12. 8-byte message buffers

Address offset (hex)	MBDSR=b00 8-byte payload
0080	MB0
0090	MB1
00A0	MB2
00B0	MB3
00C0	MB4
00D0	MB5
00E0	MB6
00F0	MB7
0100	MB8
0110	MB9
0120	MB10
0130	MB11
0140	MB12
0150	MB13
0160	MB14
0170	MB15
0180	MB16
0190	MB17
01A0	MB18
01B0	MB19
01C0	MB20
01D0	MB21
01E0	MB22
01F0	MB23
0200	MB24

Table continues on the next page...

Table 44-12. 8-byte message buffers (continued)

Address offset (hex)	MBDSR=b00 8-byte payload
0210	MB25
0220	MB26
0230	MB27
0240	MB28
0250	MB29
0260	MB30
0270	MB31
0280	MB32
0290	MB33
02A0	MB34
02B0	MB35
02C0	MB36
02D0	MB37
02E0	MB38
02F0	MB39
0300	MB40
0310	MB41
0320	MB42
0330	MB43
0340	MB44
0350	MB45
0360	MB46
0370	MB47
0380	MB48
0390	MB49
03A0	MB50
03B0	MB51
03C0	MB52
03D0	MB53
03E0	MB54
03F0	MB55
0400	MB56
0410	MB57
0420	MB58
0430	MB59
0440	MB60
0450	MB61
0460	MB62

Table continues on the next page...

Table 44-12. 8-byte message buffers (continued)

Address offset (hex)	MBDSR=b00 8-byte payload
0470	MB63
0480	MB64
0490	MB65
04A0	MB66
04B0	MB67
04C0	MB68
04D0	MB69
04E0	MB70
04F0	MB71
0500	MB72
0510	MB73
0520	MB74
0530	MB75
0540	MB76
0550	MB77
0560	MB78
0570	MB79
0580	MB80
0590	MB81
05A0	MB82
05B0	MB83
05C0	MB84
05D0	MB85
05E0	MB86
05F0	MB87
0600	MB88
0610	MB89
0620	MB90
0630	MB91
0640	MB92
0650	MB93
0660	MB94
0670	MB95

Table 44-13. 16-byte message buffers

Address offset (hex)	MBDSR=b01 16-byte payload
0080	MB0
0098	MB1
00B0	MB2
00C8	MB3
00E0	MB4
00F8	MB5
0110	MB6
0128	MB7
0140	MB8
0158	MB9
0170	MB10
0188	MB11
01A0	MB12
01B8	MB13
01D0	MB14
01E8	MB15
0200	MB16
0218	MB17
0230	MB18
0248	MB19
0260	MB20
0280	MB21
0298	MB22
02B0	MB23
02C8	MB24
02E0	MB25
02F8	MB26
0310	MB27
0328	MB28
0340	MB29
0358	MB30
0370	MB31
0388	MB32
03A0	MB33
03B8	MB34
03D0	MB35
03E8	MB36
0400	MB37

Table continues on the next page...

Table 44-13. 16-byte message buffers (continued)

Address offset (hex)	MBDSR=b01 16-byte payload
0418	MB38
0430	MB39
0448	MB40
0460	MB41
0480	MB42
0498	MB43
04B0	MB44
04C8	MB45
04E0	MB46
04F8	MB47
0510	MB48
0528	MB49
0540	MB50
0558	MB51
0570	MB52
0588	MB53
05A0	MB54
05B8	MB55
05D0	MB56
05E8	MB57
0600	MB58
0618	MB59
0630	MB60
0648	MB61
0660	MB62

Table 44-14. 32-byte message buffers

Address offset (hex)	MBDSR=b10 32-byte payload
0080	MB0
00A8	MB1
00D0	MB2
00F8	MB3
0120	MB4
0148	MB5
0170	MB6
0198	MB7

Table continues on the next page...

Table 44-14. 32-byte message buffers (continued)

Address offset (hex)	MBDSR=b10 32-byte payload
01C0	MB8
01E8	MB9
0210	MB10
0238	MB11
0280	MB12
02A8	MB13
02D0	MB14
02F8	MB15
0320	MB16
0348	MB17
0370	MB18
0398	MB19
03C0	MB20
03E8	MB21
0410	MB22
0438	MB23
0480	MB24
04A8	MB25
04D0	MB26
04F8	MB27
0520	MB28
0548	MB29
0570	MB30
0598	MB31
05C0	MB32
05E8	MB33
0610	MB34
0638	MB35

Table 44-15. 64-byte message buffers

Address offset (hex)	MBDSR=b11 64-byte payload
0080	MB0
00C8	MB1
0110	MB2
0158	MB3
01A0	MB4

Table continues on the next page...

Table 44-15. 64-byte message buffers (continued)

Address offset (hex)	MBDSR=b11 64-byte payload
01E8	MB5
0230	MB6
0280	MB7
02C8	MB8
0310	MB9
0358	MB10
03A0	MB11
03E8	MB12
0430	MB13
0480	MB14
04C8	MB15
0510	MB16
0558	MB17
05A0	MB18
05E8	MB19
0630	MB20

44.4.43 Rx FIFO structure

When the CAN_MCR[RFEN] bit is set, the memory area from 0x80 to 0xDC (which is normally occupied by MBs 0–5) is used by the reception FIFO engine.

The region 0x80-0x8C contains the output of the FIFO which must be read by the CPU as a message buffer. This output contains the oldest message that has been received but not yet read. The region 0x90-0xDC is reserved for internal use of the FIFO engine.

An additional memory area, which starts at 0xE0 and may extend up to 0x2DC (normally occupied by MBs 6–37) depending on the CAN_CTRL2[RFFN] field setting, contains the ID filter table (configurable from 8 to 128 table elements) that specifies filtering criteria for accepting frames into the FIFO.

Out of reset, the ID filter table flexible memory area defaults to 0xE0 and extends only to 0xFC, which corresponds to MBs 6 to 7 for RFFN = 0, for backward compatibility with previous versions of FlexCAN.

The following shows the Rx FIFO data structure.

Table 44-16. Rx FIFO structure

	0 3	7 8	9	10	11	12 13 14 15	16 23	24 31
0x80	IDHIT		SRR	IDE	RTR	DLC	TIME STAMP	
0x84		ID standard				ID extended		
0x88	Data byte 0		Data byte 1				Data byte 2	Data byte 3
0x8C	Data byte 4		Data byte 5				Data byte 6	Data byte 7
0x90 to 0xDC	Reserved							
0xE0	ID filter table element 0							
0xE4	ID filter table element 1							
0xE8 to 0x2D4	ID filter table elements 2 to 125							
0x2D8	ID filter table element 126							
0x2DC	ID filter table element 127							

= Unimplemented or reserved

Each ID filter table element occupies an entire 32-bit word and can be compounded by one, two, or four Identifier Acceptance Filters (IDAF) depending on the CAN_MCR[IDAM] field setting. The following figures show the IDAF indexation.

The following table shows the three different formats of the ID table elements. Note that all elements of the table must have the same format. See Rx FIFO for more information.

Table 44-17. ID table structure

	0	1	2 7	8 15	16	17	18 23	24 30	31
A	RTR	IDE	RXIDA (standard = 2–12, extended = 2–30)						
B	RTR	IDE	RXIDB_0 (standard = 2–12, extended = 2–15)		RTR	IDE	RXIDB_1 (standard = 18–28, extended = 18–31)		
C	RXIDC_0 (std/ext = 0–7)			RXIDC_1 (std/ext = 8–15)	RXIDC_2 (std/ext = 16–23)			RXIDC_3 (std/ext = 24–31)	

= Unimplemented or Reserved

RTR — Remote Frame

This bit specifies if Remote Frames are accepted into the FIFO if they match the target ID.

1 = Remote Frames can be accepted and data frames are rejected

0 = Remote Frames are rejected and data frames can be accepted

IDE — Extended Frame

Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID.

1 = Extended frames can be accepted and standard frames are rejected

0 = Extended frames are rejected and standard frames can be accepted

RXIDA — Rx Frame Identifier (Format A)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (2 to 12) are used for frame identification. In the extended frame format, all bits are used.

RXIDB_0, RXIDB_1 — Rx Frame Identifier (Format B)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (2 to 12 and 18 to 28) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 — Rx Frame Identifier (Format C)

Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

IDHIT — Identifier Acceptance Filter Hit Indicator

This 9-bit field indicates which Identifier Acceptance Filter was hit by the received message that is in the output of the Rx FIFO. See Rx FIFO for more information.

44.5 Functional description

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed by a set of Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see Message buffer structure). The memory corresponding to the first 38 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with individual mask register for up to 32 ID Filter Table elements.

For Classical CAN frames, simultaneous reception through FIFO and mailbox is supported. For CAN FD frames, reception is supported through mailboxes only. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be "active" at a given time if it can participate in both the Matching and Arbitration processes. An Rx MB with a 0b0000 code is inactive (refer to Table 44-7). Similarly, a Tx MB with a 0b1000 or 0b1001 code is also inactive (refer to Table 44-8).

The FlexCAN module is also able to receive and transmit messages in CAN FD format. The Message Buffers are sized to adequately store the quantity of data bytes selected by the MBDSRn bit fields in CAN_FDCTRL register. The quantity of FD MBs available for a given quantity of data bytes is described CAN_FDCTRL register. See also FlexCAN Memory Partition for CAN FD.

44.5.1 Transmit process

To transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the following procedure:

1. Check whether the respective interrupt bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control and Status word to request an abortion of the transmission. Wait for the corresponding IFLAG bit to be asserted by polling the CAN_IFLAG register or by the interrupt request if enabled by the respective IMASK bit. Then read back the CODE field to check if the transmission was aborted or transmitted (see Transmission abort mechanism). If backwards compatibility is desired (CAN_MCR[AEN] bit is negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the MB but then the pending frame may be transmitted without notification (see Mailbox inactivation).
3. Write the ID word.
4. Write the data bytes.

5. Write the DLC, Control, and CODE fields of the Control and Status word to activate the MB. When CAN_MCR[FDEN] is set, write also the EDL, BRS and ESI bits.

When the MB is activated, it participates in the arbitration process and is eventually transmitted according to its priority. When the DLC value stored in the MB selected for transmission is larger than the respective MB payload size, FlexCAN adds the necessary number of bytes with constant 0xCC pattern to complete the expected DLC.

At the end of the successful transmission, the value of the Free Running Timer is written into the Time Stamp field, the CODE field in the Control and Status word is updated, both CAN_CRC and CAN_FDCRC Registers are updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new CODE field after transmission depends on the code that was used to activate the MB (see Table 44-7 and Table 44-8 in Message buffer structure).

When the Abort feature is enabled (CAN_MCR[AEN] is asserted), after the Interrupt Flag is asserted for a Mailbox configured as transmit buffer, the Mailbox is blocked. Therefore the CPU is not able to update it until the Interrupt Flag is negated by CPU. This means that the CPU must clear the corresponding IFLAG bit before starting to prepare this MB for a new transmission or reception.

44.5.2 Arbitration process

The arbitration process scans the Mailboxes searching the Tx one that holds the message to be sent in the next opportunity. This Mailbox is called the *arbitration winner*.

The scan starts from the lowest number Mailbox and runs toward the higher ones.

The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the CAN_CTRL2[TASD] field value.
- During the Error Delimiter field of a CAN frame.
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the CAN bus has still not reached the first bit of the Intermission field.
- When there is CPU write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the Intermission field.

- When CHI is in Idle state and the CPU writes to the C/S word of any MB.
- When FlexCAN exits Bus Off state.
- Upon leaving Freeze mode or Low Power mode.

If the arbitration process does not manage to evaluate all Mailboxes before the CAN bus has reached the first bit of the Intermission field the temporary arbitration winner is invalidated and the FlexCAN will not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx Mailboxes at the end of the scan according to both CAN_CTRL1[LBUF] and CAN_MCR[LPRIOEN] bits settings.

44.5.2.1 Lowest-number Mailbox first

If CAN_CTRL1[LBUF] bit is asserted the first (lowest number) active Tx Mailbox found is the arbitration winner. CAN_MCR[LPRIOEN] bit has no effect when CAN_CTRL1[LBUF] is asserted.

44.5.2.2 Highest-priority Mailbox first

If CAN_CTRL1[LBUF] bit is negated, then the arbitration process searches the active Tx Mailbox with the highest priority, which means that this Mailbox's frame would have a higher probability to win the arbitration on CAN bus when multiple external nodes compete for the bus at the same time.

The sequence of bits considered for this arbitration is called the *arbitration value* of the Mailbox. The highest-priority Tx Mailbox is the one that has the lowest arbitration value among all Tx Mailboxes.

If two or more Mailboxes have equivalent arbitration values, the Mailbox with the lowest number is the arbitration winner.

The composition of the arbitration value depends on CAN_MCR[LPRIOEN] bit setting.

44.5.2.2.1 Local Priority disabled

If CAN_MCR[LPRIOEN] bit is negated the arbitration value is built in the exact sequence of bits as they would be transmitted in a CAN frame (see the following table) in such a way that the Local Priority is disabled.

Table 44-18. Composition of the arbitration value when Local Priority is disabled

Format	Mailbox Arbitration Value (32 bits)				
Standard (IDE = 0)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

44.5.2.2.2 Local Priority enabled

If Local Priority is desired CAN_MCR[LPRIOEN] must be asserted. In this case the Mailbox PRIO field is included at the very left of the arbitration value (see the following table).

Table 44-19. Composition of the arbitration value when Local Priority is enabled

Format	Mailbox Arbitration Value (35 bits)					
Standard (IDE = 0)	PRIO (3 bits)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	PRIO (3 bits)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

As the PRIO field is the most significant part of the arbitration value Mailboxes with low PRIO values have higher priority than Mailboxes with high PRIO values regardless the rest of their arbitration values.

Note that the PRIO field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

44.5.2.3 Arbitration process (continued)

After the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out and after it is done, write access to the C/S word of the corresponding MB is blocked (if the AEN bit in CAN_MCR register is asserted). Write access is restored in the following events:

- After the MB is transmitted and the corresponding IFLAG bit is cleared by the CPU
- FlexCAN enters in Freeze mode or Bus Off
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules.

Arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. CAN_CTRL2[TASD] bit value may be changed to optimize the arbitration start point.
- During CAN BusOff state from TX_ERR_CNT=124 to 128. CAN_CTRL2[TASD] bit value may be changed to optimize the arbitration start point.
- During C/S write by CPU in BusIdle. First C/S write starts arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, Tx arbitration process is pending. If there is no arbitration winner after the arbitration process has finished, then the TX arbitration machine begins a new arbitration process. If there is a pending arbitration and BusIdle state starts then an arbitration process is triggered. In this case the first and second C/S write in BusIdle will not restart the arbitration process. It is possible that there is not enough time to finish arbitration in WaitForBusIdle state and the next state is Idle. In this case the scan is not interrupted, and it is completed during BusIdle state. During this arbitration C/S write does not cause arbitration restart.
- Arbitration winner deactivation during a valid arbitration window.
- Upon exiting Freeze mode (first bit of the WaitForBusIdle state). If there is a re-synchronization during WaitForBusIdle, the arbitration process is restarted.

Arbitration process stops in the following situations:

- All Mailboxes were scanned
- A Tx active Mailbox is found in case of Lowest Buffer feature enabled
- Arbitration winner inactivation or abort during any arbitration process
- There was not enough time to finish Tx arbitration process (for instance, when a deactivation was performed near the end of frame). In this case arbitration process is pending.
- Error or Overload flag in the bus
- Low Power or Freeze mode request in Idle state

Arbitration is considered pending as described below:

- It was not possible to finish arbitration process in time
- C/S write during arbitration if write is performed in a MB whose number is lower than the Tx arbitration pointer
- Any C/S write if there is no Tx Arbitration process in progress
- Rx Match has just updated a Rx Code to Tx Code
- Entering Busoff state

C/S write during arbitration has the following effect:

- If C/S write is performed in the arbitration winner, a new process is restarted immediately.
- If C/S write is performed in a MB whose number is higher than the Tx arbitration pointer, the ongoing arbitration process will scan this MB as normal.

44.5.3 Receive process

To be able to receive CAN frames into a Mailbox, the CPU must prepare it for reception by executing the following steps:

1. If the Mailbox is active (either Tx or Rx) inactivate the Mailbox (see Mailbox inactivation), preferably with a safe inactivation (see Transmission abort mechanism).
2. Write the ID word
3. Write the EMPTY code (0b0100) to the CODE field of the Control and Status word to activate the Mailbox. No setup is required for EDL, BRS and ESI bits, they are overwritten by the respective bit fields in the received message.

After the MB is activated, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the Mailbox is updated by the *move-in* process (see Move-in) as follows:

1. The received Data field (8 bytes at most for Classical CAN message format and up to 64 bytes for CAN FD message format) is stored.
2. The received Identifier field is stored.
3. The value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the Mailbox's Time Stamp field.
4. The received SRR, IDE, RTR, EDL, BRS, ESI and DLC fields are stored.
5. The CODE field in the Control and Status word is updated (see Table 44-7 and Table 44-8 in Section Message buffer structure).
6. A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The recommended way for CPU servicing (read) the frame received in an Mailbox is using the following procedure:

1. Read the Control and Status word of that Mailbox.

2. Check if the BUSY bit is deasserted, indicating that the Mailbox is locked. Repeat step 1) while it is asserted. See Mailbox lock mechanism.
3. Read the contents of the Mailbox. Once Mailbox is locked now, its contents won't be modified by FlexCAN Move-in processes. See Move-in.
4. Acknowledge the proper flag at IFLAG registers.
5. Read the Free Running Timer. It is optional but recommended to unlock Mailbox as soon as possible and make it available for reception.

The CPU should poll for frame reception by the status flag bit for the specific Mailbox in one of the IFLAG Registers and not by the CODE field of that Mailbox. Polling the CODE field does not work because once a frame was received and the CPU services the Mailbox (by reading the C/S word followed by unlocking the Mailbox), the CODE field will not return to EMPTY. It will remain FULL, as explained in Table 44-7 . If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the Mailbox without a prior *safe inactivation*, a newly received frame matching the filter of that Mailbox may be lost.

CAUTION

In summary: *never do polling by reading directly the C/S word of the Mailboxes. Instead, read the IFLAG registers.*

Note that the received frame's Identifier field is always stored in the matching Mailbox, thus the contents of the ID field in an Mailbox may change if the match was due to masking. When CAN_MCR[SRXDIS] bit is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching Rx Mailbox, and no interrupt flag or interrupt signal will be generated. Otherwise, when CAN_MCR[SRXDIS] bit is deasserted, FlexCAN can receive frames transmitted by itself if there exists a matching Rx Mailbox.

To be able to receive CAN frames through the Rx FIFO, the CPU must enable and configure the Rx FIFO during Freeze mode (see Rx FIFO). Upon receiving the Frames Available in Rx FIFO interrupt (see the description of the BUF5I bit "Frames available in Rx FIFO" bit in the CAN_IFLAG1 register), the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (optional: needed only if a mask was used for IDE and RTR bits)
2. Read the ID field (optional: needed only if a mask was used)
3. Read the Data field

4. Read the CAN_RXFIR register (optional)
5. Clear the Frames Available in Rx FIFO interrupt by writing 1 to CAN_IFLAG1[BUF5I] bit (mandatory: releases the MB and allows the CPU to read the next Rx FIFO entry)

When CAN_MCR[DMA] is asserted, upon receiving a frame in FIFO, CAN_IFLAG1[BUF5I] generates a DMA request and does not generate a CPU interrupt (see Rx FIFO under DMA Operation). The CAN_IMASK1 bits in Rx FIFO region are not used.

The DMA controller must service the received frame using the following procedure:

1. Read the Control and Status word (read 0x80 address, optional)
2. Read the ID field (read 0x84 address, optional)
3. Read all Data Bytes (start read at 0x88 address, optional)
4. Read the last Data Bytes (read 0x8C address is mandatory)

44.5.4 Matching process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the priority of scanning can be selected between Mailboxes and FIFO filters. The matching starts from the lowest number Message Buffer toward the higher ones. If no match is found within the first structure then the other is scanned subsequently. In the event that the FIFO is full, the matching algorithm always looks for a matching MB outside the FIFO region.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- If the received frame is a remote frame, the start point is the CRC field of the frame
- If the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame
- If the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame

If a matching ID is found in the FIFO table or in one of the Mailboxes, the contents of the Rx SMB are transferred to the FIFO or to the matched Mailbox by the move-in process. If any CAN protocol error is detected then no match results are transferred to the FIFO or to the matched Mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and the active Rx Mailboxes (CODE is EMPTY, FULL, OVERRUN or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The Rx SMB has the same structure of a Mailbox. The reception structures (Rx FIFO or Mailboxes) associated with the matching elements that had a successful comparison are the *matched structures*. The *matching winner* is selected at the end of the scan among those matched structures and depends on conditions described ahead. See the following table.

Table 44-20. Matching architecture

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EACEN]	MB[IDE]	MB[RTR]	MB[ID[1]]	MB[CODE]
Mailbox	0	-	0	cmp[2]	no_cmp[3]	cmp_msk[4]	EMPTY or FULL or OVERRUN
Mailbox	0	-	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	0	-	cmp	no_cmp	cmp	RANSWER
Mailbox	1	1	0	cmp	no_cmp	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	1	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
FIFO[5]	-	-	-	cmp_msk	cmp_msk	cmp_msk	-

1. For Mailbox structure, If SMB[IDE] is asserted, the ID is 29 bits (ID Standard + ID Extended). If SMB[IDE] is negated, the ID is only 11 bits (ID Standard). For FIFO structure, the ID depends on IDAM.
2. cmp: Compares the Rx SMB contents with the MB contents regardless the masks.
3. no_cmp: The Rx SMB contents are not compared with the MB contents.
4. cmp_msk: Compares the Rx SMB contents with MB contents taking into account the masks.
5. SMB[IDE] and SMB[RTR] are not taken into account when IDAM is type C.

A reception structure is *free-to-receive* when any of the following conditions is satisfied:

- The CODE field of the Mailbox is EMPTY
- The CODE field of the Mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the CPU and unlocked as described in Mailbox lock mechanism)
- The CODE field of the Mailbox is either FULL or OVERRUN and an inactivation (see Mailbox inactivation) is performed
- The Rx FIFO is not full

The scan order for Mailboxes and Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for Mailboxes is affected by the CAN_MCR[IRMQ] bit. If it is negated, the matching winner is the first matched Mailbox regardless if it is free-to-receive or not. If it is asserted, the matching winner is selected according to the priority below:

1. the first free-to-receive matched Mailbox;
2. the last non free-to-receive matched Mailbox.

It is possible to select the priority of scan between Mailboxes and Rx FIFO by the CAN_CTRL2[MRP] bit.

If the selected priority is Rx FIFO first:

- If the Rx FIFO is a matched structure and is free-to-receive, then the Rx FIFO is the matching winner regardless of the scan for Mailboxes
- Otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched among Mailboxes as described above

If the selected priority is Mailboxes first:

- If a free-to-receive matched Mailbox is found, it is the matching winner regardless of the scan for Rx FIFO
- If no matched Mailbox is found, then the matching winner is searched in the scan for the Rx FIFO
- If both conditions above are not satisfied and a non free-to-receive matched Mailbox is found, then the matching winner determination is conditioned by the CAN_MCR[IRMQ] bit:
- - If CAN_MCR[IRMQ] bit is negated, the matching winner is the first matched Mailbox
  - If CAN_MCR[IRMQ] bit is asserted, the matching winner is the Rx FIFO if it is a free-to-receive matched structure; otherwise, the matching winner is the last non free-to-receive matched Mailbox

See the following table for a summary of matching possibilities.

Table 44-21. Matching possibilities and resulting reception structures

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
No FIFO, only MB, match is always MB first						
0	0	X[1]	None[2]	-[3]	None	Frame lost by no match
0	0	X	Free[4]	-	FirstMB	

Table continues on the next page...

Table 44-21. Matching possibilities and resulting reception structures (continued)

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
0	1	X	None	-	None	Frame lost by no match
0	1	X	Free	-	FirstMb	
0	1	X	NotFree	-	LastMB	Overrun
FIFO enabled, no match in FIFO is as if FIFO does not exist						
1	0	X	None	None[5]	None	Frame lost by no match
1	0	X	Free	None	FirstMB	
1	1	X	None	None	None	Frame lost by no match
1	1	X	Free	None	FirstMb	
1	1	X	NotFree	None	LastMB	Overrun
FIFO enabled, Queue disabled						
1	0	0	X	NotFull[6]	FIFO	
1	0	0	None	Full[7]	None	Frame lost by FIFO full (FIFO Overflow)
1	0	0	Free	Full	FirstMB	
1	0	0	NotFree	Full	FirstMB	
1	0	1	None	NotFull	FIFO	
1	0	1	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	0	1	Free	X	FirstMB	
1	0	1	NotFree	X	FirtsMb	Overrun
FIFO enabled, Queue enabled						
1	1	0	X	NotFull	FIFO	
1	1	0	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	1	0	Free	Full	FirstMB	
1	1	0	NotFree	Full	LastMb	Overrun
1	1	1	None	NotFull	FIFO	
1	1	1	Free	X	FirstMB	
1	1	1	NotFree	NotFull	FIFO	
1	1	1	NotFree	Full	LastMb	Overrun

1. This is a don't care condition.
2. Matched in MB "None" means that the frame has not matched any MB (free-to-receive or non-free-to-receive).
3. This is a forbidden condition.
4. Matched in MB "Free" means that the frame matched at least one MB free-to-receive regardless of whether it has matched MBs non-free-to-receive.
5. Matched in FIFO "None" means that the frame has not matched any filter in FIFO. It is as if the FIFO didn't exist (CAN_CTRL2[RFEN]=0).
6. Matched in FIFO "NotFull" means that the frame has matched a FIFO filter and has empty slots to receive it.

7. Matched in FIFO "Full" means that the frame has matched a FIFO filter but couldn't store it because it has no empty slots to receive it.

If a non-safe Mailbox inactivation (see Mailbox inactivation) occurs during matching process and the Mailbox inactivated is the temporary matching winner, then the temporary matching winner is invalidated. The matching elements scan is not stopped nor restarted, it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the FIFO is disabled, IRMQ is enabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm finds the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm finds MB number 2 again, but it is not "free-to-receive", so it keeps looking, finds MB number 5 and stores the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive", so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages are queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. See the description of the Rx Individual Mask Registers (CAN_RXIMRx). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is a "don't care". Note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed while the module is in Freeze mode; otherwise, they are blocked by hardware.

FlexCAN also supports an alternate masking scheme with only four mask registers (CAN_RXFGMASK, CAN_RXMGMASK, CAN_RX14MASK and CAN_RX15MASK) for backwards compatibility with legacy applications. This alternate masking scheme is enabled when the IRMQ bit in the CAN_MCR Register is negated.

44.5.5 Receive Process under Pretended Networking Mode

Pretended Networking mode adds specific wake up functionality in low power mode (Stop mode). When Pretended Network mode (PN) is enabled by asserting the PNET_EN bit in the Module Configuration Register (CAN_MCR)), FlexCAN continues processing Rx CAN messages under low power mode, able to detect specific wake up messages by filtering them against ID and payload target values using a pre-selected matching criteria. Wake up functionality is not available for messages in CAN FD format. While in Pretended Networking mode, CAN FD format messages are ignored.

PN registers are located in the 0x0B00 - 0x0B7C address range and can be written only in Freeze mode. These registers are used for writing PN configuration (both control and target values) prior entering to Pretended Networking mode, and for reading wake up flags and the received message ID and data when returning back to normal mode after wake up. The CPU must wait for CAN_MCR[LPMACK] to be negated before performing any access to FlexCAN PN registers.

PN control registers are described in Pretended Networking Control 1 Register (CAN_CTRL1_PN) and Pretended Networking Control 2 Register (CAN_CTRL2_PN). The control bit fields that configure the filtering criteria are:

- PLFS: payload filtering selection.
- IDFS: ID filtering selection.
- FCS: filtering combination selection.

PN target values are:

- FLT_IDE: IDE target value used to filter the incoming message by its format (standard or extended).
- FLT_RTR: RTR target value used to filter the incoming message by its type (data or remote frame).
- FLT_DLC_HI and FLT_DLC_LO: target DLC range used to filter the size of payload part of an incoming message.
- FLT_ID1: ID target value used to filter the incoming message ID (equal to, smaller than or equal, greater than or equal, or the lower limit value in an ID range).
- FLT_ID2: ID target value used as the upper limit in an ID range.
- PL1: Payload target value used to filter the incoming message payload (equal to, smaller than or equal, greater than or equal, or the lower limit value in a payload range).
- PL2: Payload target value used as the upper limit in a payload range.

IDE, RTR, ID and payload filters have their respective masks. These masks determine which bits are taken into account in equality comparisons ("1's" in certain mask positions) and which ones are don't care ("0's" in other mask positions). ID and payload masks are used only for exact ID and/or exact Payload comparisons.

The ID of Rx incoming messages can be filtered based on the following criteria:

- A match with the exact ID value by detecting the equality between the ID field of the incoming message and the content of target CAN_FLT_ID1 register. The ID mask is used.
- A match with the maximum range of ID, i.e. any message with ID value smaller than or equal to the content of target CAN_FLT_ID1 register is accepted. The ID mask is not used.
- A match with the minimum range of ID, i.e. any message with ID value greater than or equal to the content of target CAN_FLT_ID1 register is accepted. The ID mask is not used.
- A match inside a range of IDs, i.e. any message with an ID value that is greater than or equal to the content of target CAN_FLT_ID1 register and smaller than or equal to the content of target CAN_FLT_ID2_IDMASK register is accepted. The ID mask is not used.

See CAN_CTRL1_PN[IDFS] in Pretended Networking Control 1 Register (CAN_CTRL1_PN).

The above criteria for ID filtering must be coherent with FLT_IDE and FLT_RTR target values in CAN_FLT_ID1 register. Only Rx frames that match the respective IDE and RTR bits to the contents of FLT_IDE and FLT_RTR bit fields will be compared. When range of IDs is selected (CAN_CTRL1_PN[IDFS] = 11), both FLT_ID1 and FLT_ID2 are referred to the same FLT_IDE and FLT_RTR bits in CAN_FLT_ID1 register.

The ID mask is applied only to the exact ID comparison filtering option (CAN_CTRL1_PN[IDFS] = 00) to determine which bits are taken into account in the comparison. For the exact match option, the mask can select any bit within the ID field. For maximum range, minimum range and inside range comparisons, the ID mask is not considered.

The IDE and RTR masks are applied in both exact and range ID comparison filtering options to determine which bits are taken into account in comparison.

Similarly to the ID criteria, 64-bit data or payloads (PL) of Rx incoming messages can be filtered based on the following criteria:

- A match with the exact payload value by detecting the equality between the payload field of the incoming message and the content of PL1 register. The payload mask is used.

- A match with the maximum range of payload, i.e. any message with payload value smaller than or equal to the content of PL1 register is accepted. The payload mask is not used.
- A match with the minimum range of payload, i.e. any message with payload value greater than or equal to the content of PL1 register is accepted. The payload mask is not used.
- A match inside a range of payloads, i.e. any message with a payload value that is greater than or equal to the content of PL1 register and smaller than or equal to the content of PL2 register is accepted. The payload mask is not used.

See CAN_CTRL1_PN[PLFS] in Pretended Networking Control 1 Register (CAN_CTRL1_PN).

The above criteria for payload filtering must be coherent with FLT_DLC upper and lower limit values in CAN_FLT_DLC register. The payload of a Rx incoming message is filtered in accordance to the selected criteria only if the DLC value of the Rx incoming message is inside a DLC range:

- greater than or equal to the FLT_DLC_LO (lower limit) and
- lower than or equal to the FLT_DLC_HI (upper limit).

Conversely, a DLC value out of the specified range results in mismatch. By making FLT_DLC_LO = FLT_DLC_HI, only payloads of specified quantity of bytes will be filtered. DLC is not maskable.

When the inside range of payloads option is selected (CAN_CTRL1_PN[PLFS] = 11), both PL1 and PL2 are considered with the 8-byte data length. All the data bytes excluded by the DLC of the received message are considered with value zero.

Payload mask is only used in the exact match option (CAN_CTRL1_PN[PLFS] = 00) to select which bits or bytes in the 8-byte data field of both Rx incoming message and the contents of PL1 register are selected for matching. Mask length must be in accordance to the expected range of DLC values. For maximum range, minimum range and inside range comparisons, the payload mask is not considered.

When a remote frame is received by FlexCAN and the CAN_CTRL1[FCS] bit is configured to select the payload comparison, the payload filtering is not considered and the comparison results in a mismatch.

Rx incoming messages can also be filtered based upon the quantity and rate of message reception, specifically:

- Several messages that match the filtering criteria for ID or payload a predefined quantity of times. This quantity can be configured in the 1 to 255 range. See the Pretended Networking Control 1 Register (CAN_CTRL1_PN).
- No message matching the filtering criteria for ID or payload up to a timeout trigger. That is, non-reception of a matching message for a defined quantity of time. See the Pretended Networking Control 2 Register (CAN_CTRL2_PN).

FlexCAN can generate a wake up timeout event from an internal timer with associated comparator circuitry capable to generate a timeout flag when the counting reaches the pre-defined timeout value, as specified in CAN_CTRL2_PN[MATCHTO].

The above filtering criteria can be used together as follows:

- Message ID filtering only.
- Message ID filtering and Payload filtering.
- Message ID filtering only occurring N times.
- Message ID filtering and Payload filtering occurring N times.

The timeout counter runs concurrently with the reception filtering process. Both engines, timeout counter and message filtering, are independent. If an incoming message matches the selected filter criteria, the timeout counter keeps counting until the CPU wakes-up. Conversely, if the timeout counter reaches the target value then the message filtering process continue to filter incoming messages until the CPU wakes-up. The CAN_WU_MTC[MCOUNTER] field will report the number of matched messages occurred under Pretended Networking mode up to the moment the CPU wakes up.

Under Pretended Networking mode, the wake up event that may occur will set the respective wake up flag (see Pretended Networking Wake Up Match Register (CAN_WU_MTC)):

- In case of a successful matching in accordance to the selected filtering criteria, the CAN_WU_MTC[WUMF].
- In case of a timeout trigger, the CAN_WU_MTC[WTOF].

Any of these flags will generate interrupts to the CPU, provided the respective mask bits are enabled (WUMF_MSK or WTOF_MSK in CAN_CTRL1_PN register).

There are four WMB's (Wake up Message Buffers) used to store incoming messages in Pretended Networking mode. Up to four messages can be stored (see Pretended Networking Wake Up Message Buffer Registers (CAN_WMB0 - CAN_WMB3)). When CTRL1_PN[NMATCH] value is 1, just one message is received if matching the filtering criteria, and this message is stored in CAN_WMB0. If NMATCH value is between 2 and 4, CAN_WMB1, CAN_WMB2 and CAN_WMB3 are used to store the second,third and fourth matching messages, respectively. If NMATCH is greater than 4, the last four matching messages are stored in the WMBs, respecting the WMB index to indicate the

arrival order, the latest is stored in CAN_WMB3. Only the valid data bytes of the incoming match message is stored in data field of WMBs. The non-valid data bytes is read as zero. In case of DLC=0 and RTR=1 the data field is filled with zero. In any of the above cases, the wake up interrupt is generated just when the filtering criteria is completed and CAN_CTRL1_PN[WUMF_MSK] is enabled.

When a non-match wake-up event occurs (timeout or external) and MCOUNTER register is equal or greater than 4, the message stored in WMB0 does not have a valid content. The CAN_WMB0 is used as buffer for the current message in CAN bus. Messages received during Pretended Networkig mode don't have timestamps and respective field in the WMB structure must be ignored.

Under low power mode(Stop), all processes are shut down except for the PN functionality inside CAN_PE sub-block, that is kept clocked by the Oscillator clock (see Clock domains and restrictions). FlexCAN continues to receive Rx incoming messages and just compares them against the predefined target values and in accordance to the selected filtering criteria. The matching, arbitration, move-in and move-out processes, normally available in Normal mode, are not performed under Pretended Networking mode.

The FlexCAN under Pretended Networking reacts to messages on the CAN bus in the same manner as in Normal mode (i.e. generates acknowledge bits, detect and count errors, etc.).

44.5.6 Move process

There are two types of move process: move-in and move-out.

44.5.6.1 Move-in

The move-in process is the copy of a message received by an Rx SMB to a Rx Mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the CAN_RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described ahead. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (see Matching process) and all of the following conditions are true:

- The CAN bus has reached or let past either:
  - The second bit of Intermission field next to the frame that carried the message that is in the Rx SMB
  - The first bit of an overload frame next to the frame that carried the message that is in the Rx SMB

- There is no ongoing matching process
- The destination Mailbox is not locked by the CPU
- There is no ongoing move-in process from another Rx SMB. If more than one move-in processes are to be started at the same time both are performed and the newest substitutes the oldest.

The term *pending move-in* is used throughout the documentation and stands for a move-to-be that still does not satisfy all of the aforementioned conditions.

The move-in is cancelled and the Rx SMB is able to receive another message if any of the following conditions is satisfied:

- The destination Mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished
- There is a previous pending move-in to the same destination Mailbox
- The Rx SMB is receiving a frame transmitted by the FlexCAN itself and the self-reception is disabled (CAN_MCR[SRXDIS] bit is asserted)
- Any CAN protocol error is detected

Note that the pending move-in is not cancelled if the module enters Freeze or Low-Power mode. It only stays on hold waiting for exiting Freeze and Low-Power mode and to be unlocked. If an MB is unlocked during Freeze mode, the move-in happens immediately.

The move-in process is the execution by the FlexCAN of the following steps:

1. Push IDHIT into the RXFIR FIFO if the message is destined to the Rx FIFO.
2. Read all data words from the Rx SMB in accordance to the selected payload size for the Rx storage element.
3. Write all data words to the Rx Mailbox in accordance to the selected payload size for the Rx storage element. If the data size of the storage element is smaller than the original payload size described in the message's DLC field, the payload is truncated and the high order bytes that do not fit the destination size are lost.
4. Read the Control/Status and ID words from the Rx SMB.
5. Write Control/Status and ID words to the Rx Mailbox, and update the CODE field.

The move-in process is not atomic, in such a way that it is immediately cancelled by the inactivation of the destination Mailbox (see Mailbox inactivation) and in this case the Mailbox may be left partially updated, thus incoherent. The exception is if the move-in destination is an Rx FIFO Message Buffer, then the process cannot be cancelled.

The BUSY Bit (least significant bit of the CODE field) of the destination Message Buffer is asserted while the move-in is being performed to alert the CPU that the Message Buffer content is temporarily incoherent.

44.5.6.2 Move-out

The move-out process is the copy of the content from a Tx Mailbox to the Tx SMB when a message for transmission is available (see Section "Arbitration process"). The move-out occurs in the following conditions:

- The first bit of Intermission field
- During Bus Off state when TX Error Counter is in the 124 to 128 range
- During Bus Idle state
- During Wait For Bus Idle state

The move-out process is not atomic. Only the CPU has priority to access the memory concurrently out of Bus Idle state. In Bus Idle, the move-out has the lowest priority to the concurrent memory accesses.

44.5.7 Data coherence

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in Transmit process and Receive process.

44.5.7.1 Transmission abort mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

Two primary conditions must be fulfilled in order to abort a transmission:

- CAN_MCR[AEN] bit must be asserted
- The first CPU action must be the writing of abort code (0b1001) into the CODE field of the Control and Status word.

Active MBs configured for transmission must be aborted first before they can be updated. If the abort code is written to a Mailbox that is currently being transmitted or to a Mailbox that was already loaded into the Tx SMB for transmission, the write operation is blocked and the transmission is not disturbed. However, the abort request is captured and kept pending until one of the following conditions is satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into Freeze mode

- The module enters the BusOff state
- There is an overload frame

If none of the conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the IFLAG register, and an interrupt to the CPU is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. On the other hand, if one of the above conditions is reached, the frame is not transmitted; therefore, the abort code is written into the CODE field, the interrupt flag is set in the IFLAG, and an interrupt is (optionally) generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, then the write operation is not blocked; therefore, the MB is updated and the interrupt flag is set. In this way the CPU just needs to read the abort code to make sure the active MB was *safely inactivated*. Although the AEN bit is asserted and the CPU wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One Mailbox is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

- CPU checks the corresponding IFLAG and clears it, if asserted.
- CPU writes 0b1001 into the CODE field of the C/S word.
- CPU waits for the corresponding IFLAG indicating that the frame was either transmitted or aborted.
- CPU reads the CODE field to check if the frame was either transmitted (CODE=0b1000) or aborted (CODE=0b1001).
- It is necessary to clear the corresponding IFLAG in order to allow the MB to be reconfigured.
- It is necessary to reconfigure the EDL, BRS, and ESI bits of the aborted MB before transmitting it again.

44.5.7.2 Mailbox inactivation

Inactivation is a mechanism provided to protect the Mailbox against updates by the FlexCAN internal processes, thus allowing the CPU to rely on Mailbox data coherence after having updated it, even in Normal mode.

Inactivation of transmission Mailboxes must be performed just when MCR[AEN] bit is deasserted.

If a Mailbox is inactivated, it participates in neither the arbitration process nor the matching process until it is reactivated. See Transmit process and Receive process for more detailed instructions on how to inactivate and reactivate a Mailbox.

To inactivate a Mailbox, the CPU must update its CODE field to INACTIVE (either 0b0000 or 0b1000).

Because the user is not able to synchronize the CODE field update with the FlexCAN internal processes, an inactivation can have the following consequences:

- A frame in the bus that matches the filtering of the inactivated Rx Mailbox may be lost without notice, even if there are other Mailboxes with the same filter
- A frame containing the message within the inactivated Tx Mailbox may be transmitted without setting the respective IFLAG

In order to perform a *safe inactivation* and avoid the above consequences for Tx Mailboxes, the CPU must use the Transmission Abort mechanism (see Transmission abort mechanism).

The inactivation automatically unlocks the Mailbox (see Mailbox lock mechanism).

NOTE

Message Buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on the FIFO region by FlexCAN. CPU must maintain data coherency in the FIFO region when RFEN is asserted.

44.5.7.3 Mailbox lock mechanism

Other than Mailbox inactivation, FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the Control and Status word of an Rx MB with codes FULL or OVERRUN, FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation, and therefore it sets an internal lock flag for that MB. The lock is released when the CPU reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB regardless of its code. A CPU write into the C/S word also unlocks the MB, but this procedure is not recommended for normal unlock use because it cancels a pending-move and potentially may lose a received message. The MB locking prevents a new frame from being written into the MB while the CPU is reading it.

NOTE

The locking mechanism applies only to Rx MBs that are not part of the FIFO and have a code different than INACTIVE (0b0000) or EMPTY[1] (0b0100). Also, Tx MBs can not be locked.

1. In previous FlexCAN versions, reading the C/S word locked the MB even if it was EMPTY. This behavior is maintained when the IRMQ bit is negated.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no "free-to-receive" MBs, so it decides to override MB number 5. However, this MB is locked, so the new message can not be written there. It will remain in the Rx SMB waiting for the MB to be unlocked, and only then will be written to the MB.

If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the Rx SMB and there will be no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved-in from the Rx SMB to the MB, the BUSY bit on the CODE field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

Note

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Inactivation takes precedence over locking. If the CPU inactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the Rx SMB will not be transferred anymore to the MB. An MB is unlocked when the CPU reads the Free Running Timer Register (see Section "Free Running Timer Register (CAN_TIMER)"), or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during a low power mode (see Modes of operation), and it takes place only when the module resumes to Normal or Freeze modes.

44.5.8 Rx FIFO

The Rx FIFO is receive-only and is enabled by asserting the CAN_MCR[RFEN] bit. The reset value of this bit is zero to maintain software backward compatibility with previous versions of the module that did not have the FIFO feature.

CAUTION

Rx FIFO must not be enabled when CAN FD feature is enabled.

The FIFO is 6-message deep. The memory region occupied by the FIFO structure (both Message Buffers and FIFO engine) is described in Rx FIFO structure. The CPU can read the received messages sequentially, in the order they were received, by repeatedly reading a Message Buffer structure at the output of the FIFO.

The CAN_IFLAG1[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, the CPU can read the message (accessing the output of the FIFO as a Message Buffer) and the CAN_RXFIR register and then clear the interrupt. If there are more messages in the FIFO the act of clearing the interrupt updates the output of the FIFO with the next message and update the CAN_RXFIR with the attributes of that message, reissuing the interrupt to the CPU. Otherwise, the flag remains negated. The output of the FIFO is only valid whilst the CAN_IFLAG1[BUF5I] is asserted.

The CAN_IFLAG1[BUF6I] (Rx FIFO Warning) is asserted when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. The flag remains asserted until the CPU clears it.

The CAN_IFLAG1[BUF7I] (Rx FIFO Overflow) is asserted when an incoming message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. The flag remains asserted until the CPU clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted too.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to CAN_CTRL2[RFFN] setting, that can be configured to one of the following formats (see also Rx FIFO structure):

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)

- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

Note

A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can read in the IDHIT field from C/S word, as shown in the Rx FIFO Structure description. Another way the CPU can obtain this information is by accessing the CAN_RXFIR register. The CAN_RXFIR[IDHIT] field refers to the message at the output of the FIFO and is valid while the CAN_IFLAG1[BUF5I] flag is asserted. The CAN_RXFIR register must be read only before clearing the flag, which guarantees that the information refers to the correct frame within the FIFO.

Up to 32 elements of the filter table are individually affected by the Individual Mask Registers (CAN_RXIMRx), according to the setting of CAN_CTRL2[RFFN], allowing very powerful filtering criteria to be defined. If the CAN_MCR[IRMQ] bit is negated, then the FIFO filter table is affected by CAN_RXFGMASK.

44.5.8.1 Rx FIFO under DMA Operation

The receive-only FIFO can support DMA, this feature is enabled by asserting both the CAN_MCR[RFEN] and CAN_MCR[DMA] bits. The reset value of CAN_MCR[DMA] bit is zero to maintain backward compatibility with previous versions of the module that did not have the DMA feature.

The DMA controller can read the received message by reading a Message Buffer structure at the FIFO output port at the 0x80-0x8C address range.

When CAN_MCR[DMA] is asserted the CPU must not access the FIFO output port address range. Before enabling the CAN_MCR[DMA], the CPU must service the IFLAGs asserted in the Rx FIFO region. Otherwise, these IFLAGs may show that the FIFO has data to be serviced, and mistakenly generate a DMA request. Before disabling the CAN_MCR[DMA], the CPU must perform a clear FIFO operation.

The CAN_IFLAG1[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO, consequently a DMA request is generated simultaneously. Upon receiving the request, the DMA controller can read the message (accessing the output of the FIFO as a Message Buffer). The DMA reading process must end by reading address 0x8C, which clears the CAN_IFLAG1[BUF5I] and

updates both the FIFO output with the next message (if FIFO is not empty) and the CAN_RXFIR register with the attributes of the new message. If there are more messages stored in the FIFO, the CAN_IFLAG1[BUF5I] will be re-asserted and another DMA request is issued. Otherwise, the flag remains negated.

NOTE

CAN_RXFIR register contents cannot be read after DMA completes the FIFO read. The IDHIT information is also available in the C/S word at address 0x080 (see Rx FIFO structure.

The CAN_IFLAG1[BUF6I] and CAN_IFLAG1[BUF7I] are not used when the DMA feature is enabled.

When FlexCAN is working with DMA, the CPU does not receive any Rx FIFO interruption and must not clear the related IFLAGs. In addition, the related IMASKs are not used to mask the generation of DMA requests.

44.5.8.2 Clear FIFO Operation

When CAN_MCR[RFEN] is asserted, the clear FIFO operation is a feature used to empty FIFO contents. With CAN_MCR[RFEN] asserted the Clear FIFO occurs when the CPU writes 1 in CAN_IFLAG1[BUF0I]. This operation can only be performed in Freeze Mode and is blocked by hardware in other modes. This operation does not clear the FIFO IFLAGs, consequently the CPU must service all FIFO IFLAGs before execute the clear FIFO task.

When Rx FIFO is working with DMA, the clear FIFO operation clears the CAN_IFLAG1[BUF5I] and the DMA request is canceled.

CAUTION

Clear FIFO operation does not clear IFLAGs, except when CAN_MCR[DMA] is asserted, in this case only the CAN_IFLAG1[BUF5I] is cleared.

44.5.9 CAN protocol related features

This section describes the CAN protocol related features.

44.5.9.1 CAN FD ISO compliance

The CAN FD protocol has been improved to increase the failure detection capability that was in the original CAN FD protocol, which is also called non-ISO CAN FD by CAN in Automation (CiA). A three-bit stuff counter and a parity bit have been introduced in the improved CAN FD protocol, now called ISO CAN FD. The CRC calculation has also been modified. All these improvements make the ISO CAN FD protocol incompatible with the non-FD CAN FD protocol. The non-ISO CAN FD is still supported by FlexCAN so that it can be used mainly during an intermediate phase, for evaluation and development purposes.

Therefore, it is strongly recommended to configure FlexCAN to the ISO CAN FD protocol by setting the ISOCANFDEN field in the CAN_CTRL2 register.

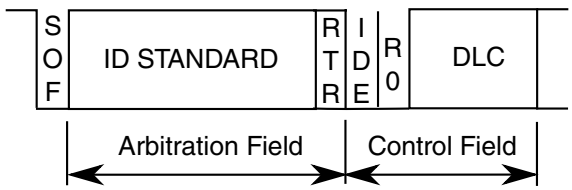
44.5.9.2 CAN FD frames

The ISO 11898-1 standard specifies the Classical Frame format compliant to ISO 11898-1 (2003) and introduces the CAN Flexible Data Rate Frame format. The Classical Frame format allows bit rates up to 1 Mbit/s and payloads up to 8 bytes per frame. The Flexible Data Rate Frame format allows bit rates higher than 1 Mbit/s and payloads longer than 8 bytes per frame. FlexCAN can receive and transmit CAN FD messages interleaved with Classical CAN messages.

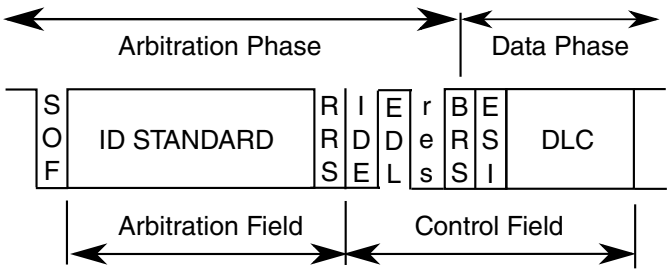
There are three additional control bits in the CAN FD frame. The Extended Data Length (EDL) bit enables a longer data payload with different data length coding. The Bit Rate Switch (BRS) bit decides whether the bit rate is switched inside a CAN FD format frame. The Error State Indicator (ESI) flag is transmitted dominant by error active nodes, and recessive by error passive nodes. There is no Remote Frames (see Remote frames) in the CAN FD format. A message configured to transmit a Remote Frame is always sent out in the Classical CAN format. When a FD frame is received and matches a mailbox, the RTR bit in the receiving message buffer is negated. The RTR bit must be considered in classical frames only.

CAN FD messages may be formatted as long frames where the data field exceeds 8 bytes, and may range from 12 up to 64 bytes. They can also be configured to support bit rate switching, where the control field, the data field, and the CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame. Messages in Classical CAN format are limited to transport a maximum payload of 8 bytes at nominal rate. The following figure illustrates the message formats for Classical and FD frames with either standard or extended ID.

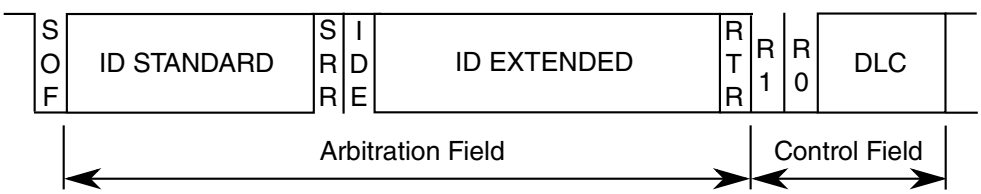
CAN Standard Format



CAN FD Standard Format



CAN Extended Format



CAN FD Extended Format

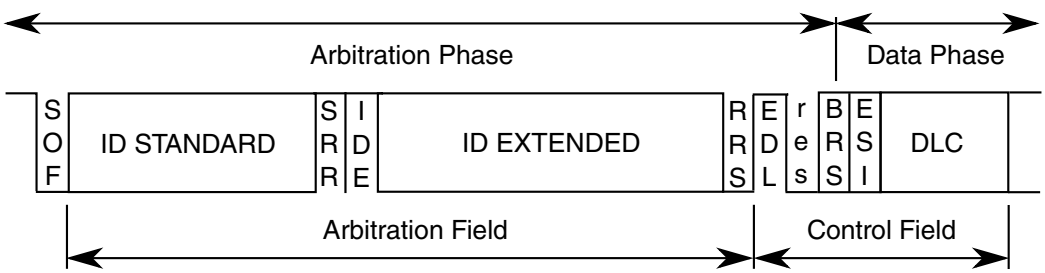


Figure 44-2. CAN message formats

The ability to receive and transmit CAN FD messages is enabled by the CAN_MCR[FDEN] bit. Either a recessive R0 bit in CAN frames with 11-bit identifiers or a recessive R1 bit in CAN frames with 29-bit identifiers are decoded as an EDL bit (not a reserved one). A CAN FD frame is recognized by a recessive EDL bit, while a

Classical CAN frame is recognized by a dominant EDL bit. The BRS bit specifies whether this frame switches the bit rate in its data phase. A long frame is decoded in accordance to the DLC field value (see DLC definition in Message buffer structure).

CAN FD messages can be transmitted with two different bit rates. The first part of a CAN FD frame, from the Start Of Frame (SOF) bit until the Bit Rate Switch (BRS) bit, also called the arbitration phase, is transmitted with the nominal bit rate based on a set of nominal CAN bit timing configuration values. The second part, from the BRS bit until the CRC Delimiter bit, also named the data phase, is transmitted with the data bit rate defined by a second set of CAN data bit timing configuration values. Finally, from the CRC Delimiter until the Intermission bits, the transmission resumes to nominal bit rate. In CAN FD frames with bit rate switching, the bit timing is changed inside the frame at the sample point of the BRS bit if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the CAN_CBT register (also by CAN_CTRL1 register for backward compatibility). Upon detecting a recessive BRS bit, the CAN data bit timing is used as defined by the CAN_FDCBT register.

NOTE

If the length of the time quantum in the nominal bit timing and the length of the time quantum in the data bit timing are not identical, a quantization error of up to one time quantum of the arbitration phase may be present as a phase error. This situation can occur after the switch from arbitration to data phase and will last until the next synchronization event. Thus, the length of the time quantum should be the same in nominal and data bit timing in order to minimize the chance of error frames on the CAN bus, and to optimize the clock tolerance in networks that use FD frames.

CAN_FDCTRL[FDRATE] enables the transmission of all frames with bit rate switching if the BRS bit in the selected Tx MB is set. If FDRATE is negated, the transmission is performed at nominal rate regardless of the BRS bit value. The CAN_FDCTRL[FDRATE] bit can be written any time but takes effect only for the next message transmitted or received.

The nominal bit timing is resumed at either the sample point of the CRC Delimiter bit or when an error is detected, whichever occurs first. The following figure describes the mechanism for entering and leaving the data phase when BRS bit is recessive.

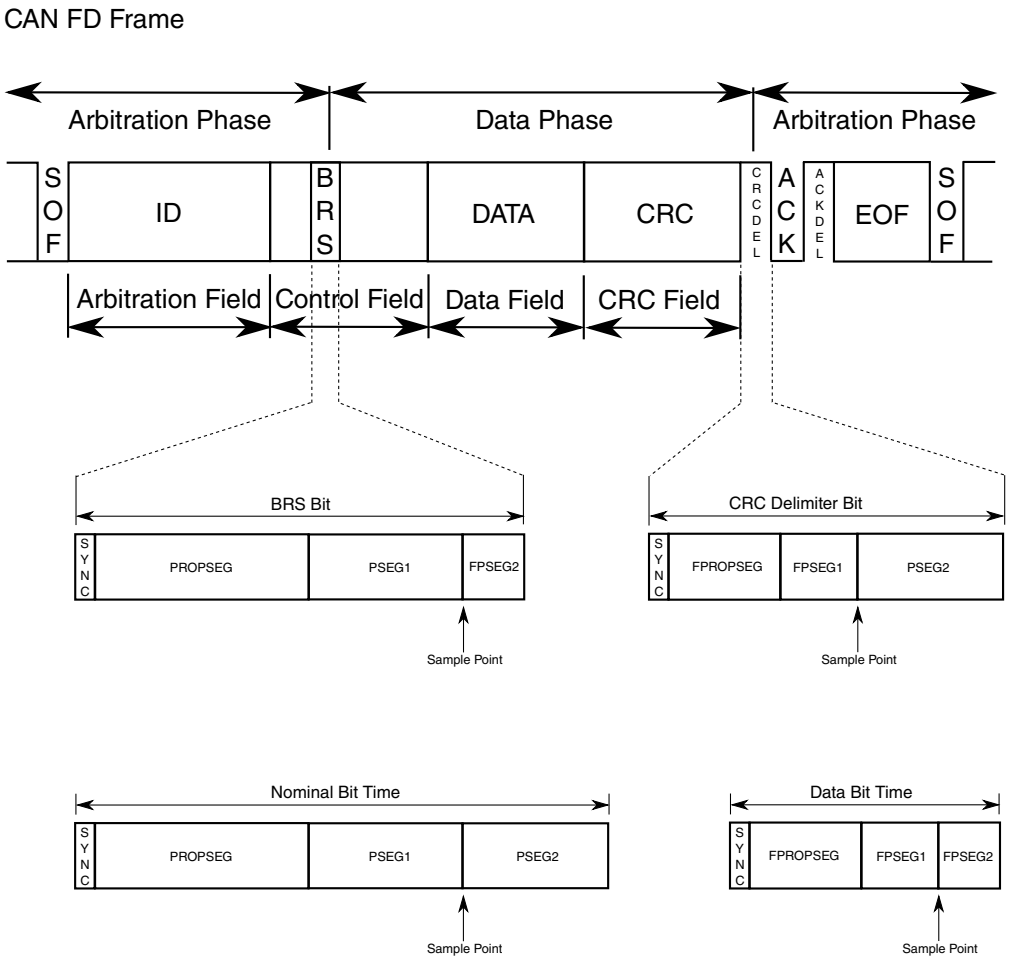


Figure 44-3. Bit rate switching mechanism for CAN FD messages

NOTE

In Classical CAN frames, the CRC delimiter is one single recessive bit. In CAN FD frames, the CRC delimiter may consist of one or two recessive bits. FlexCAN sends only one recessive bit as the CRC delimiter, but it accepts two recessive bits before the edge from recessive to dominant that starts the acknowledge slot. As a receiver, FlexCAN sends its acknowledge bit after the first CRC delimiter bit. In CAN FD frames, FlexCAN accepts a two-bit dominant ACK slot as a valid ACK to compensate for phase shifts between the receivers.

The maximum configurable bit rate in the CAN FD data phase depends on the clock frequency of CAN_PE sub-block. For example, with a CAN_PE clock frequency of 40MHz and the shortest configurable bit time of 8 time quanta, the bit rate in the data phase is 5 Mbit/s.

The value of the ESI bit is determined either by the transmitter's error state at the start of the transmission, if the frame is originated in the FlexCAN node, or by the original transmitting node in case FlexCAN is acting as a gateway for the message. If the transmitter is error passive, ESI is transmitted recessive; otherwise, it is transmitted dominant.

There are different CRC polynomials for different CAN frame formats. The first polynomial, CRC_15, is used for all frames in Classical CAN format. The second, CRC_17, is used for frames in CAN FD format with a data field up to sixteen bytes long. The third, CRC_21, is used for frames in CAN FD format with a data field longer than sixteen bytes. Each polynomial results in a Hamming Distance of 6. At the start of the frame, all three CRC polynomials are calculated concurrently. The CRC sequence to be transmitted is selected by the values of the EDL bit and the DLC bit field. When receiving a message, FlexCAN decodes EDL and DLC to select the adequate CRC polynomial to check for a CRC error.

In CAN FD format frames, stuff bits are included in the bit stream for CRC calculation. In Classical CAN format frames, stuff bits are not included. After the transmission of the last bit relevant to the CRC calculation, the CAN_FDCRC register stores the calculated CRC for the transmitted message, with the adequate length in accordance to the type of message, for both CAN FD and non-FD messages. The CAN_CRCR register reports a valid CRC for Classical CAN messages only.

In CAN FD format frames, the CAN bit stuffing method is changed for the CRC sequence so that the stuff bits are inserted at fixed positions. When FlexCAN is transmitting a CAN FD frame, a fixed stuff bit is inserted just before the first bit of the CRC sequence, even if the last bits of the preceding field do not fulfill the CAN stuff condition. Additional stuff bits are inserted after each fourth bit of the CRC sequence. The value of any fixed stuff bit is the inverse value of its preceding bit. When FlexCAN is receiving a CAN FD frame, it discards the fixed stuff bits from the bit stream for the CRC check. A Stuff Error is detected if the fixed stuff bit has the same value as its preceding bit.

FlexCAN detects errors in CAN FD frames the same way as in Classical CAN frames. The error counters RXERRCNT and TXERRCNT in the CAN_ECR register accumulate the counts of Rx and Tx errors, respectively, for both FD and non-FD frames indistinctly. There are two extra error counters (RXERRCNT_FAST and TXERRCNT_FAST) that accumulate Rx and Tx errors occurring in the data phase of CAN FD frames with the BRS bit set only. The rules for updating the error counters are the same for both CAN FD and non-FD frames (see CAN_ECR register).

Error Flags BITERR1, BITERR0, ACKERR, CRCERR, FRMERR and STFERR in the ESR1 register report errors in both CAN FD and non-FD frames. They also generate the ERRINT interrupt if CAN_CTRL1[ERRMSK] is asserted. The CAN_ESR1 register has

additional error flags (BITERR1_FAST, BITERR0_FAST, CRCERR_FAST, FRMERR_FAST and STFERR_FAST) to individually indicate the occurrence of errors in the data phase of CAN FD frames with the BRS bit set. There is no ACKERR detected in the data phase of a CAN FD frame. Fault confinement status reported in CAN_ESR1[FLTCONF] is the same for both CAN FD and Classical CAN frames, and is based on RXERRCNT and TXERRCNT error counters only. Information contained in RXERRCNT_FAST and TXERRCNT_FAST counters may be considered as status to help detect the error nature related to the bit rate value.

When FlexCAN is in the data phase, either transmitting or receiving a CAN FD message, and detects an error, it immediately switches back to the arbitration phase and to the nominal rate to start an Error Flag.

Resynchronization and Hard Synchronization occur in CAN FD frames in the same way as in Classical CAN ones. Additionally, a Hard Synchronization is also performed at the recessive to dominant edge from EDL to R0 in CAN FD format frames. FlexCAN does not resynchronize while transmitting in the CAN FD data phase.

44.5.9.3 Transceiver Delay Compensation

The CAN FD protocol allows the transmission and reception of data at a higher bit rate than the nominal rate used in the arbitration phase when the message's BRS bit is set. This feature enables the use of rates up to 8 Mbps.

During the data phase of a CAN FD frame, the Transmitter detects a bit error if it cannot receive its own latest transmitted bit at the sample point of that bit. When bit rate switching is enabled (BRS bit is asserted), the length of the CAN bit time in the data phase can become shorter than the transceiver's loop delay, thus impeding the correct comparison between the transmitted bit and the received bit within the current CAN bit time interval.

FlexCAN supports an optional Transceiver Delay Compensation (TDC) mechanism that defines a secondary sample point where the transmitted bit is correctly compared with the received bit in order to check for bit errors.

The TDC mechanism can be enabled by the CAN_FDCTRL[TDCEN] bit and is effective only during the data phase of FD frames having the BRS bit set. It has no effect either on non-FD frames, or on FD frames transmitted at normal bit rate. The TDC is active from the sample point of the BRS bit until the sample point of the CRC Delimiter bit, provided the respective message under transmission has the BRS bit set. When it is active, a comparison is done between the real received bit and the delayed transmitted bit, where the delay is calculated based on the measured transceiver loop delay.

NOTE

The actual value of the CRC Delimiter bit is disregarded by transmitters using the Transceiver Delay Compensation mechanism. A global error at the end of the CRC Field will cause the receivers to send error frames that the transmitter will detect during Acknowledge or End of Frame.

For every transmitted FD frame having the BRS bit set, the delay measurement is triggered by the transition from the recessive EDL bit to the dominant R0 bit (as shown in the next figure). The loop delay is measured in Protocol Engine (PE) clock periods (CANCLK, see Protocol timing), from the transmitted EDL-R0 edge to the received EDL-R0 edge. The position of the secondary sample point is defined by the measured loop delay time added to an offset value specified in CAN_FDCTRL[TDCOFF]. CAN_FDCTRL[TDCVAL] bit field stores the result of this calculation. The TDCVAL value saturates at its maximum value of 15 CANCLK when the delay measurement is too long.

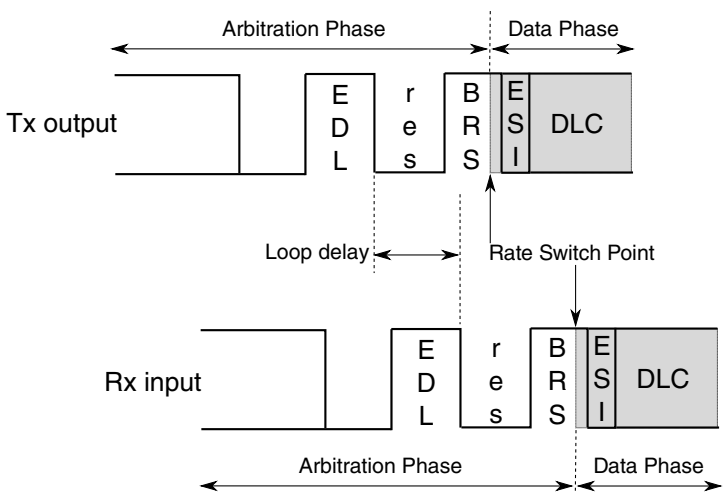


Figure 44-4. Transceiver loop delay measurement

The measured loop delay is not enough to be used to define the secondary sample point because it relates to the CAN bit edges. The transceiver delay compensation offset TDCOFF is used to shift the secondary sample point from the edge to an intermediate point inside the bit time (e.g. half of the bit time in the data phase), far away from its edges. Therefore, the TDCOFF value cannot be larger than the CAN bit duration in the data phase.

During the data phase of CAN FD frames with bit rate switching enabled, at the onset of every Tx CAN bit, the transmitted Tx bit value is temporarily stored in a buffer and a time countdown based on TDCVAL is started which ends with the comparison of the

received Rx bit (delayed by the external loop delay plus the specified offset) with the stored Tx bit. If a bit error is detected at the secondary sample point, the FlexCAN issues an error flag to the CAN bus at the next sample point.

During the arbitration phase the delay compensation is always disabled. The maximum delay which can be compensated by the FlexCAN's transceiver delay compensation during the data phase is 3 CAN bit times - 2 Tq. Beyond this limit, the CAN_FDCTRL[TDCFAIL] flag is set to indicate when the Transceiver Delay Compensation mechanism is out of range, unable to compensate the transceiver loop delay.

44.5.9.4 Remote frames

Remote frame is a special kind of frame. The user can program a mailbox to be a Remote Request Frame by configuring the mailbox as Transmit with the RTR bit set to '1'. After the remote request frame is transmitted successfully, the mailbox becomes a Receive Message Buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, it can be treated in three ways, depending on Remote Request Storing (CTRL2[RRS]) and Rx FIFO Enable (MCR[RFEN]) bits:

- If RRS is negated the frame's ID is compared to the IDs of the Transmit Message Buffers with the CODE field 0b1010. If there is a matching ID, then this mailbox frame will be transmitted. Note that if the matching mailbox has the RTR bit set, then FlexCAN will transmit a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. In the case that a remote request frame is received and matches a mailbox, this message buffer immediately enters the internal arbitration process, but is considered as a normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.
- If RRS is asserted the frame's ID is compared to the IDs of the receive mailboxes with the CODE field 0b0100, 0b0010 or 0b0110. If there is a matching ID, then this mailbox will store the remote frame in the same fashion of a data frame. No automatic remote response frame will be generated. The mask registers are used in the matching process.
- If RFEN is asserted FlexCAN will not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the CPU. Note that for

filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID). Remote Request Frames are considered as normal frames, and generate a FIFO overflow when a successful reception occurs and the FIFO is already full.

NOTE

There is no remote frame in the CAN FD format. The RTR bit is replaced by a fixed dominant RRS bit. FlexCAN receives and transmits remote frames in the Classical CAN format.

44.5.9.5 Overload frames

FlexCAN does transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

44.5.9.6 Time stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of "move-in" in the TIME STAMP field, providing network behavior with respect to time.

When the TIMER_SRC bit in CAN_CTRL2 register is asserted, the Free Running Timer is continuously clocked by an external time tick.

When the TIMER_SRC bit in CAN_CTRL2 register is negated, the Free Running Timer is clocked by the FlexCAN bit-clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate.

The Free Running Timer is not incremented during Disable, Stop, and Freeze modes. It can be reset upon a specific frame reception, enabling network time synchronization. See the TSYN description in Control 1 Register (CAN_CTRL1).

44.5.9.7 Protocol timing

The following figure shows the structure of the clock generation circuitry that feeds the CAN Protocol Engine (PE) submodule. The clock source bit CLKSRC in the CAN_CTRL1 Register defines whether the internal clock is connected to the output of a crystal oscillator (Oscillator Clock) or to the Peripheral Clock. In order to guarantee reliable operation, the clock source should be selected while the module is in Disable Mode (MDIS bit set in the Module Configuration Register).

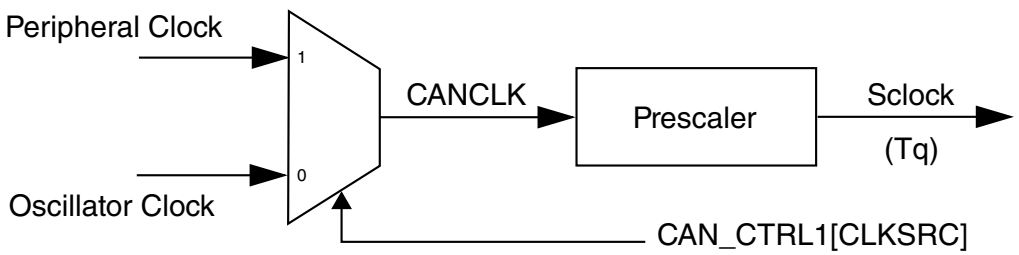


Figure 44-5. CAN engine clocking scheme

The oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The crystal oscillator clock has better jitter performance than the peripheral clock.

The FlexCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control 1 Register (CAN_CTRL1) has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW.

The CAN Bit Timing register (CAN_CBT) extends the range of the CAN bit timing variables in CAN_CTRL1. The CAN FD Bit Timing register (CAN_FDCBT) provides a second set of CAN bit timing variables to be applied at the data phase of CAN FD frames with the Bit Rate Switch (BRS) set.

NOTE

When the CAN FD feature is enabled, always set CAN_CBT[BTF] and configure the CAN bit timing variables in CAN_CBT. See CAN Bit Timing Register (CAN_CBT).

The PRESDIV field (as well as its extended range EPRESDIV and FDPRESDIV for the data phase bits of CAN FD messages) defines the Prescaler Value (see the equation below) that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum (Tq) is the atomic unit of time handled by the CAN engine.

$$Tq = \frac{(\text{PRESDIV} + 1)}{f_{CANCLK}}$$

The bit rate, which defines the rate the CAN message is either received or transmitted, is given by the formula:

CAN Bit Time = (Number of Time Quanta in 1 bit time) * Tq

$$\text{Bit Rate} = \frac{1}{\text{CAN Bit Time}}$$

A bit time is subdivided into three segments[1] (see Figure 44-6, Figure 44-7 and Table 44-22):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section
- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CAN_CTRL1 Register so that their sum (plus 2) is in the range of 2 to 16 time quanta. When CAN_CBT[BTF] bit is asserted, FlexCAN uses EPROPSEG and EPSEG1 fields from CAN_CBT register so that their sum (plus 2) is in the range of 2 to 96 time quanta. For messages in CAN FD format with the BRS bit set, FlexCAN uses FDPROPSEG and FDPSEG1 from CAN_FDCBT instead, so that their sum (plus 1) is in the range of 2 to 39 time quanta.
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CAN_CTRL1 Register (plus 1) to be 2 to 8 time quanta long. When CAN_CBT[BTF] bit is asserted, FlexCAN uses EPSEG2 fields of CAN_CBT register so that its value (plus 1) is in the range of 2 to 32 time quanta. For messages in CAN FD format with the BRS bit set, FlexCAN uses FDPSEG2 from CAN_FDCBT instead, so that its value (plus 1) is in the range of 2 to 8 time quanta. The Time Segment 2 cannot be smaller than the Information Processing Time (IPT), which value is 2 time quanta in FlexCAN.

NOTE

The bit time defined by the above time segments must not be smaller than 5 time quanta. For bit time calculations, use an

1. For further explanation of the underlying concepts, see ISO 11898-1. See also the CAN 2.0A/B protocol specification for bit timing.

Information Processing Time (IPT) of 2, which is the value implemented in the FlexCAN module.

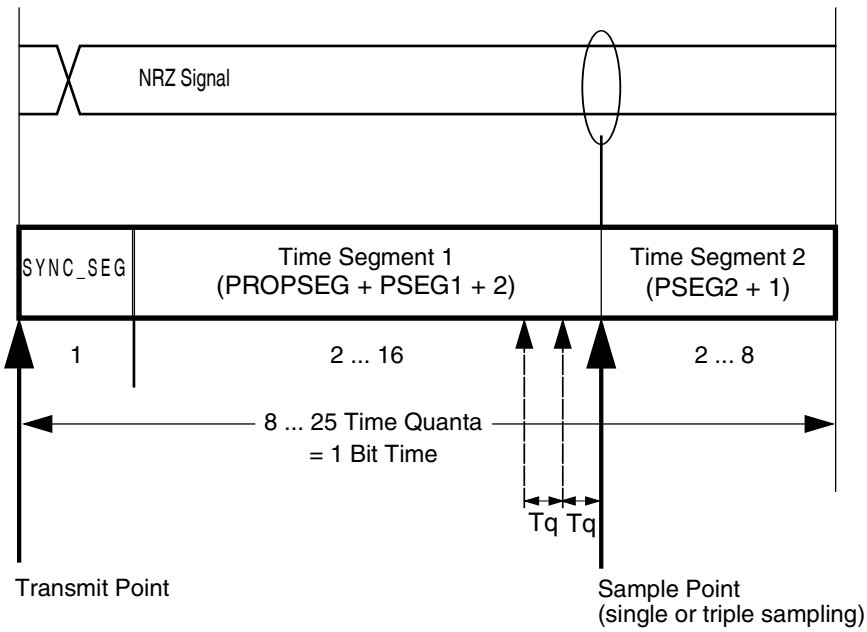


Figure 44-6. Segments within the bit time (example using CAN_CTRL1 bit timing variables for Classical CAN format)

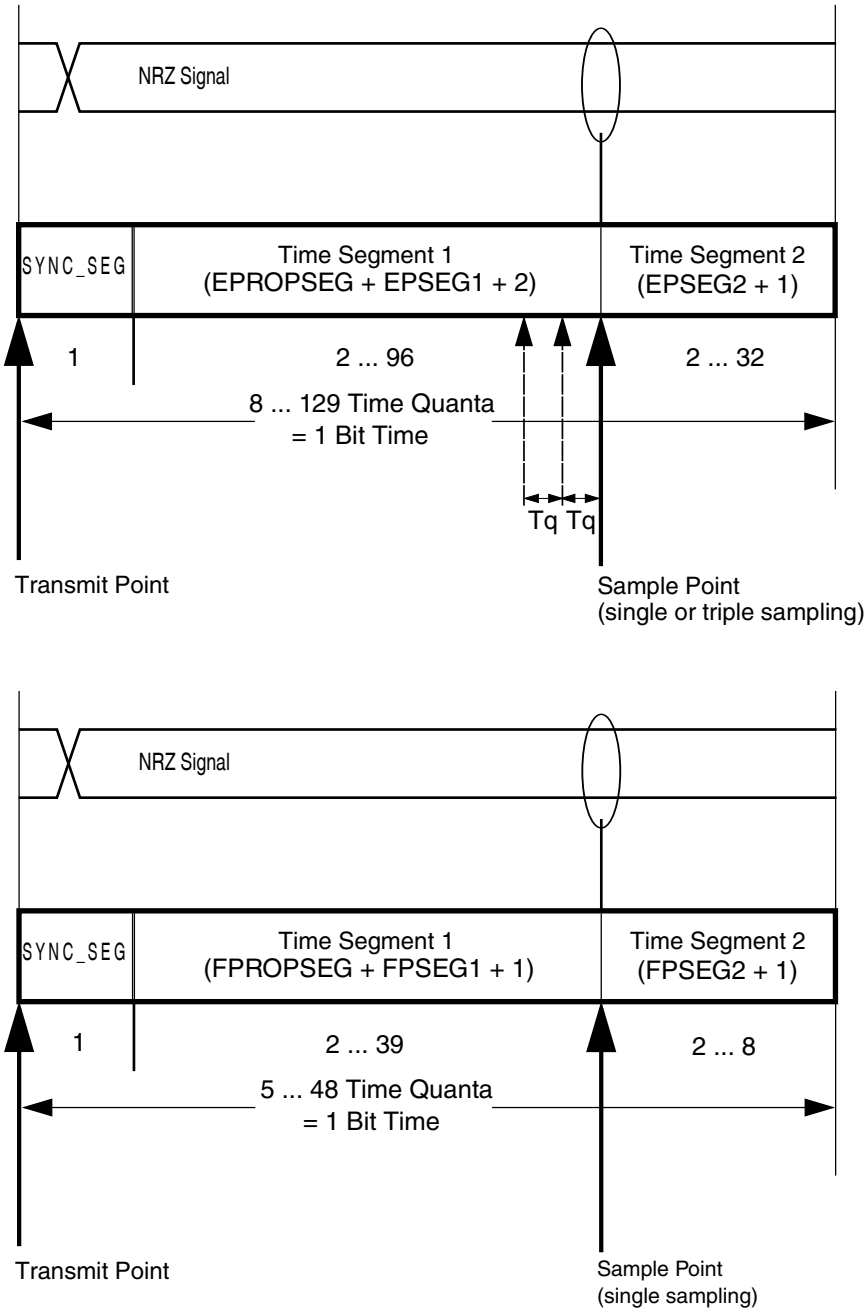


Figure 44-7. Segments within the bit time (example using CAN_CBT and CAN_FDCBT bit timing variables for CAN FD format)

Table 44-22. Time segment syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
TSEG1	Corresponds to the sum of PROPSEG and PSEG1.
TSEG2	Corresponds to the PSEG2 value.

Table continues on the next page...

Table 44-22. Time segment syntax (continued)

Syntax	Description
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

The following table gives some examples of the CAN compliant segment settings for Classical CAN format (Bosch CAN 2.0B) (non-FD) messages.

Table 44-23. Bosch CAN 2.0B standard compliant bit time segment settings

Time segment 1	Time segment 2	Re-synchronization jump width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

Note

The user must ensure the bit time settings are in compliance with the CAN Protocol standard (ISO 11898-1).

Whenever CAN bit is used as a measure of time duration (e.g. estimating the occurrence of a CAN bit event in a message), the number of peripheral clocks in one CAN bit (NumClkBit) can be calculated as:

$$\text{NumClkBit} = \frac{f_{SYS}}{f_{CANCLK}} \text{ X } (\text{PRESDIV} + 1) \text{ X } (\text{PROPSEG} + \text{PSEG1} + \text{PSEG2} + 4)$$

where:

- NumClkBit is the number of peripheral clocks in one CAN bit;
- f_{CANCLK} is the Protocol Engine (PE) Clock (see Figure "CAN Engine Clocking Scheme"), in Hz;
- f_{SYS} is the frequency of operation of the system (CHI) clock, in Hz;
- PSEG1 is the value in CAN_CTRL1[PSEG1] field;
- PSEG2 is the value in CAN_CTRL1[PSEG2] field;
- PROPSEG is the value in CAN_CTRL1[PROPSEG] field;
- PRESDIV is the value in CAN_CTRL1[PRESDIV] field.

The formula above is also applicable to the alternative CAN bit timing variables described in the CAN Bit Timing Register (CAN_CBT) and also to the CAN FD Bit Timing Register (CAN_FDCBT).

For example, 180 CAN bits = (180 x NumClkBit) peripheral clock periods.

44.5.9.8 Arbitration and matching timing

During normal reception and transmission, the matching, arbitration, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

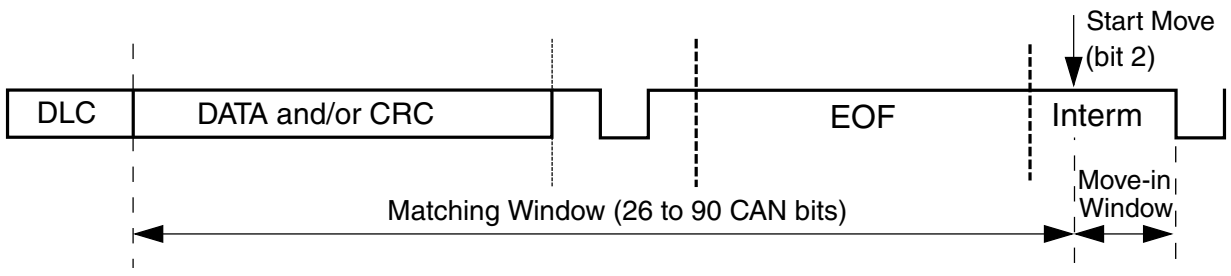


Figure 44-8. Matching and move-in time windows

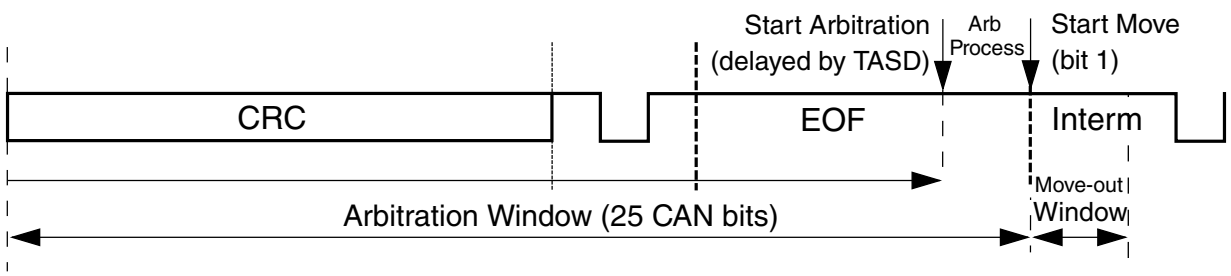


Figure 44-9. Arbitration and move-out time windows

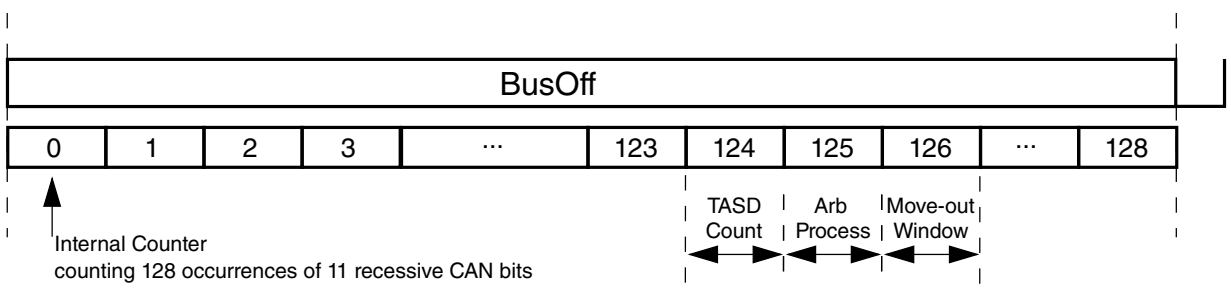


Figure 44-10. Arbitration at the end of bus off and move-out time windows

NOTE

In the preceding figures, the matching and arbitration timing does not take into account the delay caused by the concurrent

memory access due to the CPU or other internal FlexCAN sub-blocks.

44.5.9.9 Tx Arbitration start delay

The Tx Arbitration Start Delay (TASD) bit field in Control 2 register (CAN_CTRL2[TASD]) is a variable that indicates the number of CAN bits used by FlexCAN to delay the Tx Arbitration process start point from the first bit of CRC field of the current frame. This variable can be written only in Freeze mode because it is blocked by hardware in other modes.

The transmission performance is impacted by the ability of the CPU to reconfigure Message Buffers (MBs) for transmission after the end of the internal Arbitration process, where FlexCAN finds the winner MB for transmission (see Arbitration process). If the Arbitration ends too early before the first bit of Intermission field, then there is a chance that the CPU reconfigures some Tx MBs and the winner MB is no longer the best candidate to be transmitted.

TASD is useful to optimize the transmission performance by defining the Arbitration start point, as shown in the next figure, based on factors such as:

- The peripheral-to-oscillator clock ratio
- CAN bit timing variables that determine the CAN bit rate
- The number of Message Buffers (MBs) in use by the Matching and Arbitration processes.

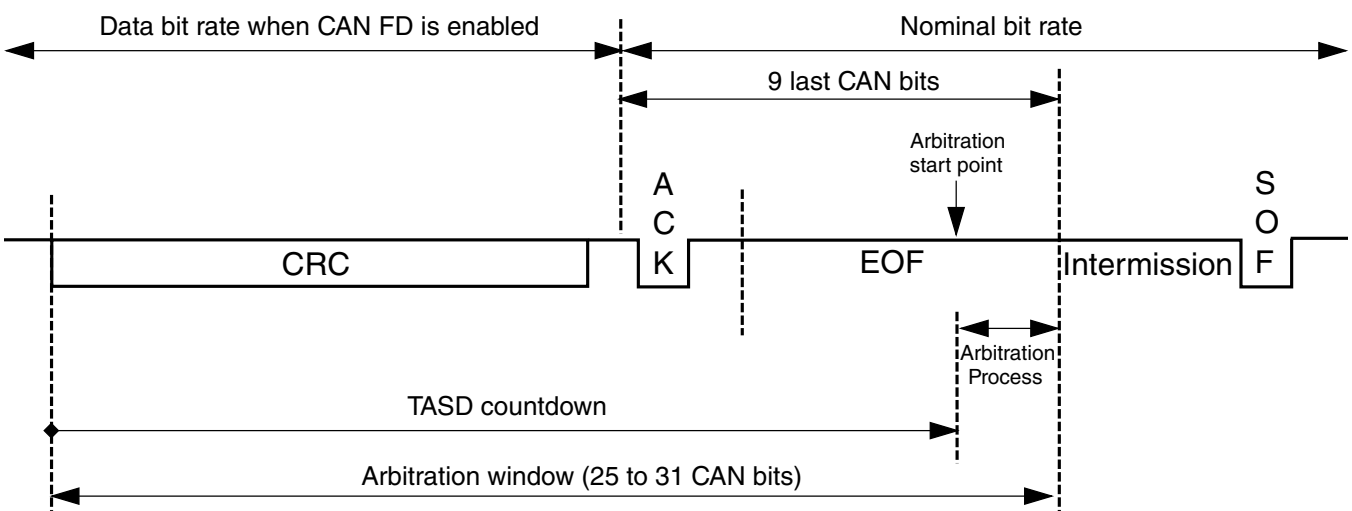


Figure 44-11. Optimal Tx Arbitration start point

The duration of an Arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and to the CAN bit rate, and inversely proportional to the peripheral clock frequency.

The optimal Arbitration timing is that in which the last MB is scanned right before the first bit of the Intermission field of a CAN frame. For instance, if there are few MBs and the peripheral/oscillator clock ratio is high and the CAN baud rate is low, then the Arbitration can be placed closer to the frame's end, adding more delay to its start point, and vice-versa.

If TASD is set to 0 then the Arbitration start is not delayed and more time is reserved for Arbitration. On the other hand, if TASD is close to 24 then the CPU can configure a Tx MB later and less time is reserved for Arbitration. If too little time is reserved for Arbitration the FlexCAN may be not be able to find a winner MB in time to be transmitted with the best chance to win the bus arbitration against external nodes on the CAN bus.

The optimal TASD value can be calculated as follows:

For CAN FD frames and $(MAXMB + 1) \leqslant NMB_{END}$

$$TASD = 31 - \frac{2 * (MAXMB + 1) + 4}{CPCB_N}$$

For CAN FD frames and $(MAXMB + 1) > NMB_{END}$

$$TASD = 22 - \frac{2 * (MAXMB + 1) - NMB_{END}}{CPCB_F}$$

For non-FD frames

$$TASD = 25 - \frac{2 * (MAXMB + 1) + 4}{CPCB}$$

where:

$$NMB_{END} = \frac{(9 * CPCB_N) - 4}{2}$$

$$BITRATE_N = \left(\frac{f_{CANCLK}}{[1 + (EPSEG1 + 1) + (EPSEG2 + 1) + (EPROPSEG + 1)] \times (EPRESDIV + 1)} \right)$$

$$BITRATE_F = \left(\frac{f_{CANCLK}}{[1 + (FPSEG1 + 1) + (FPSEG2 + 1) + FPROPSEG] \times (FPRESDIV + 1)} \right)$$

$$CPCB_N = \frac{f_{SYS}}{BITRATE_N}$$

$$CPCB_F = \frac{f_{SYS}}{BITRATE_F}$$

$$CPCB = CPCB_N$$

- MAXMB is the value in CAN_CTRL1[MAXMB] field
- NMB_{END} is the number of Message Buffers that can be scanned by the Arbitration process during the 9 last CAN bits at the end of a frame, see the figure above
- $BITRATE_N$ is the CAN bit rate in bits per second calculated by the nominal CAN bit time variables
- $BITRATE_F$ is the CAN bit rate in bits per second calculated by the data CAN bit time variables
- $CPCB_N$ is the number of peripheral clocks per CAN bit in nominal bit rate for CAN FD frames
- $CPCB_F$ is the number of peripheral clocks per CAN bit in data bit rate for CAN FD frames
- CPCB is the number of peripheral clocks per CAN bit for non-FD frames
- f_{CANCLK} is the oscillator clock, in Hz
- f_{SYS} is the peripheral clock, in Hz
- EPSEG1 is the value in CAN_CBT[EPSEG1] field (CAN_CTRL1[PSEG1] can also be used)
- EPSEG2 is the value in CAN_CBT[EPSEG2] field (CAN_CTRL1[PSEG2] can also be used)
- EPROPSEG is the value in CAN_CBT[EPROPSEG] field (CAN_CTRL1[PROPSEG] can also be used)

- EPRESDIV is the value in CAN_CBT[EPRESDIV] field (CAN_CTRL1[PRESDIV] can also be used)
- FPSEG1 is the value in CAN_FDCBT[FPSEG1] field
- FPSEG2 is the value in CAN_FDCBT[FPSEG2] field
- FPROPSEG is the value in CAN_FDCBT[FPROPSEG] field
- FPRESDIV is the value in CAN_FDCBT[FPRESDIV] field

See also Protocol timing for more details.

The following tables give the TASD value calculated for some configuration cases.

Case 1:

- Clock ratio = 2:1 (example: peripheral clock 80 MHz and oscillator clock 40 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 44-24. TASD values:

Number of Message Buffers	TASD value	Maximum Bit Rate in Data Phase (Mbaud)
16	24	Invalid
32	24	8.0
64	23	8.0
96	22	8.0

Case 2:

- Clock ratio = 1:1 (example: peripheral clock 40 MHz and oscillator clock 40 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 44-25. TASD values:

Number of Message Buffers	TASD value	Maximum Bit Rate in Data Phase (Mbaud)
16	24	Invalid
32	23	6.67
54	22	5.0
64	21	3.33
96	20	1.6

Case 3:

- Clock ratio = 2:1 (example: peripheral clock 40 MHz and oscillator clock 20 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 44-26. TASD values:

Number of Message Buffers	TASD value	Maximum Bit Rate in Data Phase (Mbaud)
16	24	Invalid
32	23	4.0
54	22	4.0
64	21	3.33
96	20	1.54

44.5.10 Clock domains and restrictions

The FlexCAN module has two clock domains asynchronous to each other:

- The Bus Domain feeds the Control Host Interface (CHI) submodule and is derived from the peripheral clock.
- The Oscillator Domain feeds the CAN Protocol Engine (PE) submodule and is derived directly from a crystal oscillator clock, so that very low jitter performance can be achieved on the CAN bus.

When CAN_CTRL1[CLKSRC] bit is set, synchronous operation occurs because both domains are connected to the peripheral clock (creating a 1:1 ratio between the peripheral and oscillator clocks).

When the two domains are connected to clocks with different frequencies and/or phases, there are restrictions on the frequency relationship between the two clock domains. In the case of asynchronous operation, the Bus Domain clock frequency must always be greater than the Oscillator Domain clock frequency.

NOTE

Asynchronous operation with a 1:1 ratio between peripheral and oscillator clocks is not allowed.

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the time slot of one CAN frame, comprised of a number of CAN bits. In order to have sufficient time to do that, the following requirements must be observed:

- The peripheral clock frequency can not be smaller than the oscillator clock frequency
- There must be a minimum number of peripheral clocks per CAN bit, as specified in the table shown below

Table 44-27. Minimum number of peripheral clocks per CAN bit for Classical CAN format

Number of Mailboxes	Value of CAN_MCR[RFEN] bit	Minimum number of peripheral clocks per CAN bit
16	0	16
32	0	16
64	0	25
96	0	37
16	1	16
32	1	17
64	1	30
96	1	42

For classical frame format, the minimum number of peripheral clocks per CAN bit specified in the preceding table determines the minimum peripheral clock frequency for a given number of Mailboxes and for an expected CAN bit rate. The CAN bit rate depends on the number of time quanta in a CAN bit, that can be defined by adjusting one or more of the bit timing values contained in either the Control 1 Register (CAN_CTRL1) or CAN Bit Time register (CAN_CBT). The time quantum (Tq) is defined in Protocol timing. The minimum number of time quanta per CAN bit must be 8; therefore, the oscillator clock frequency should be at least 8 times the CAN bit rate.

For CAN FD frame format, there are some constraints that need to be satisfied. The number of peripheral clocks per CAN bit in nominal bit rate (NumClkNomBit) can be calculated by the equation below.

$$\text{NumClkNomBit} = \frac{f_{SYS}}{f_{CANCLK}} \text{ X } (\text{PRESDIV} + 1) \text{ X } (\text{PROPSEG} + \text{PSEG1} + \text{PSEG2} + 4)$$

$$= \frac{f_{SYS}}{\text{NomBitRate}}$$

where PRESDIV, PSEG1 and PSEG2 are CAN bit time values in CTRL1 register. Alternatively, EPRESDIV, EPSEG1 and EPSEG2 values in CBT register can be used instead. NumClkNomBit can also be calculated as a function of the expected nominal bit rate used in the Arbitration Phase (NomBitRate) as shown in the equation above.

The number of CAN bits in the Data Phase of a FD Frames with the BRS bit set (fast CAN bits, in short) depends on the number of data bytes in the payload. The number of fast CAN bits (NumOfFastBits) can be determined in the table below. The less the

number of data bytes, the less the number of fast CAN bits, and less time is available for FlexCAN to scan the whole Message Buffer memory during the internal matching and arbitration processes.

Table 44-28. Number of fast CAN bits in a CAN FD frame

Minimum number of data bytes	DLC field	NumOfFastBits
0	0x0	21
1	0x1	29
2	0x2	37
3	0x3	45
4	0x4	53
5	0x5	61
6	0x6	69
7	0x7	77
8	0x8	85
12	0x9	117
16	0xA	149
20	0xB	186
24	0xC	218
32	0xD	282
48	0xE	410
64	0xF	538

The critical part of a CAN FD frame is during the Data Phase, where the CAN bit rate is faster than in the Arbitration Phase. The minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated to guarantee that enough time is available for FlexCAN to scan the Message Buffer memory during reception and transmission. The equation below calculates this constraint.

$$\text{MinNumClkFastBit}_A = \frac{(\ 8.5 \times \text{MaxNumOfMb}\ ) + 64 - (\ 9 \times \text{NumClkNomBit}\ )}{\text{NumOfFastBits}}$$

where MaxNumOfMb is the maximum number of available Mailboxes defined in CAN_MCR[MAXMB].

The clock domain crossing circuit between the CHI and PE sub-blocks also imposes a minimum number of peripheral clocks per fast CAN bit for the handshake mechanism to work properly without loosing status information through the interface, as shown in the equation below.

$$MinNumClkFastBit_B = 3 \times \left\langle 1 + \frac{f_{SYS}}{f_{CANCLK}} \right\rangle$$

Therefore, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) is determined by the larger of the two values calculated above.

$$MinNumClkFastBit = Maximum\ (\ MinNumClkFastBit_A, MinNumClkFastBit_B)$$

Then, the maximum CAN bit rate in the Data Phase of CAN FD frames (DataBitRateMAX) can be calculated as below.

$$DataBitRate_{MAX} = \frac{f_{CANCLK}}{ROUNDUP\left\langle \frac{MinNumClkFastBit \times f_{CANCLK}}{f_{SYS}} \right\rangle}$$

The peripheral and oscillator clock frequencies, the maximum number of mailboxes and the expected nominal bit rate affect the maximum data bit rate attainable by FlexCAN in CAN FD mode. Besides, the data bit rate depends on the minimum payload size of FD frames used in a given application.

To illustrate how the CAN FD bit rate is affected by the configuration of FlexCAN variables, an application example with the peripheral and oscillator clock frequencies set to 50 MHz and 40 MHz, respectively, is considered.

Step 1 - Considering the nominal bit rate as 1 Mbps, the number of peripheral clocks per CAN bit in nominal bit rate is calculated as below.

$$NumClkNomBit = \frac{50 \times 10^6}{1 \times 10^6} = 50$$

Step 2 - The number of fast CAN bits (NumOfFastBits) is determined in the table presented above. For example, if the minimum payload in FD frames is 8 bytes, then there are 85 CAN bits in the Data Phase.

Step 3 - Assuming the maximum number of mailboxes is 96, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated.

$$\text{MinNumClkFastBit}_A = \frac{(8.5 \text{ X } 96) + 64 - (9 \text{ X } 50)}{85} = 5.06$$

$$\text{MinNumClkFastBit}_B = 3 \text{ x } \left(1 + \frac{50}{40} \right) = 6.75$$

$$\text{MinNumClkFastBit} = \text{Maximum } (5.06, 6.75) = 6.75$$

Step 4 - The maximum CAN bit rate in the Data Phase can be finally found.

$$\text{DataBitRate}_{MAX} = \frac{40 \text{ x } 10^6}{\text{ROUNDUP} \left(\frac{6.75 \text{ x } 40 \text{ x } 10^6}{50 \text{ x } 10^6} \right)} = 6.667 \text{ Mbps}$$

As demonstrated in this example, even though the oscillator clock frequency (40 MHz) is adequate to generate a data rate of 8 Mbps in CAN FD mode, the specific FlexCAN configuration limits this rate to 6.667 Mbps. This limitation is mainly due to the low peripheral clock frequency that imposes the MinNumClkFastBitB bound.

The table below shows the maximum data rate for CAN FD according to clock frequencies, payload size and number of available mailboxes. See in this table that, for some cases, if the number of available mailboxes is reduced, the FlexCAN can then achieve a data rate up to 8 Mbps.

Table 44-29. Maximum CAN bit rate in Data Phase on CAN FD frames

Peripheral clock frequency (MHz)	Payload size	Number of available mailboxes	Maximum data rate (Mbps)
40	8	94	6.667
40	8	114	5.0
40	12	117	6.667
40	12	128	5.714

Table continues on the next page...

Table 44-29. Maximum CAN bit rate in Data Phase on CAN FD frames (continued)

Peripheral clock frequency (MHz)	Payload size	Number of available mailboxes	Maximum data rate (Mbps)
50	12 to 64	128	6.667
60	8	126	8.0
60	12	128	8.0
67	6	128	8.0
80	3	128	8.0
100	0	128	8.0

44.5.11 Modes of operation details

The FlexCAN module has functional modes and low-power modes. See Modes of operation for an introductory description of all the modes of operation. The following sub-sections contain functional details on Freeze mode and the low-power modes.

CAUTION

"Permanent Dominant" failure on CAN Bus line is not supported by FlexCAN. If a Low-Power request or Freeze mode request is done during a "Permanent Dominant", the corresponding acknowledge can never be asserted.

44.5.11.1 Freeze mode

This mode is requested either by the CPU through the assertion of the HALT bit in the CAN_MCR Register or when the chip is put into Debug mode. In both cases it is also necessary that the FRZ bit is asserted in the CAN_MCR Register and the module is not in a low-power mode.

The acknowledgement is obtained through the assertion by the FlexCAN of FRZ_ACK bit in the same register. The CPU must only consider the FlexCAN in Freeze mode when both request and acknowledgement conditions are satisfied.

When Freeze mode is requested, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in does not prevent going to Freeze mode.
- Ignores the Rx input pin and drives the Tx pin as recessive

- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in CAN_MCR

After requesting Freeze mode, the user must wait for the FRZ_ACK bit to be asserted in CAN_MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible, except for CAN_CTRL1[CLKSRC] bit that can be read but cannot be written.

Exiting Freeze mode is done in one of the following ways:

- CPU negates the FRZ bit in the CAN_MCR Register
- The chip is removed from Debug Mode and/or the HALT bit is negated

The FRZ_ACK bit is negated after the protocol engine recognizes the negation of the freeze request. When out of Freeze mode, FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

44.5.11.2 Module Disable mode

This low power mode is normally used to temporarily disable a complete FlexCAN block, with no power consumption. It is requested by the CPU through the assertion of the CAN_MCR[MDIS] bit, and the acknowledgement is obtained through the assertion by the FlexCAN of the CAN_MCR[LPMACK] bit. The CPU must only consider the FlexCAN in Disable mode when both request and acknowledgement conditions are satisfied.

If the module is disabled during Freeze mode, it requests to disable the clocks to the PE and CHI sub-modules, sets the LPMACK bit and negates the FRZACK bit.

It is not recommended to use Module Disable mode under Pretended Networking mode. Negate the MDIS bit and wait for LPMACK to negate before setting CAN_MCR[PNET_EN].

If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.

- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the clocks to the PE and CHI sub-modules
- Sets the NOTRDY and LPMACK bits in CAN_MCR

The Bus Interface Unit continues to operate, enabling the CPU to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, the Rx FIFO Global Mask Register. The Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the module is in Disable Mode. Exiting from this mode is done by negating the MDIS bit by the CPU, which causes the FlexCAN to request to resume the clocks and negate the LPMACK bit after the CAN protocol engine recognizes the negation of disable mode requested by the CPU.

44.5.11.3 Stop mode

This is a system low-power mode in which all chip clocks can be stopped for maximum power savings. The Stop mode is globally requested by the CPU and the acknowledgement is obtained through the assertion by the FlexCAN of a Stop Acknowledgement signal. The CPU must only consider the FlexCAN in Stop mode when both request and acknowledgement conditions are satisfied.

If FlexCAN receives the global Stop mode request during Freeze mode, it sets the LPMACK bit, negates the FRZACK bit and then sends the Stop Acknowledge signal to the CPU, in order to shut down the clocks globally.

If Stop mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Sets the NOTRDY and LPMACK bits in CAN_MCR
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Stop mode is exited when the CPU resumes the clocks and removes the Stop Mode request. This can be as a result of the Self Wake mechanism.

In the Self Wake mechanism, if the SLFWAK bit in CAN_MCR Register was set at the time FlexCAN entered Stop mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN sets the WAKINT bit in the CAN_ESR Register and, if enabled by the WAKMSK bit in CAN_MCR, generates a Wake Up interrupt to the CPU. Upon receiving the interrupt, the CPU should resume the clocks and remove the Stop mode request. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up. The following table details the effect of SLFWAK and WAKMSK upon wake-up from Stop mode. Note that wake-up from Stop mode only works when both bits are asserted.

After the CAN protocol engine recognizes the negation of the Stop mode request, the FlexCAN negates the LPMACK bit. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up.

Table 44-30. Wake-up from Stop Mode

SLFWAK	WAKINT	WAKMSK	Chip clocks enabled	Wake-up interrupt generated
0	-	-	No	No
0	-	-	No	No
1	0	0	No	No
1	0	1	No	No
1	1	0	No	No
1	1	1	Yes	Yes

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the Rx CAN input line while in Stop mode. See the WAKSRC bit in in the description of the Module Configuration Register (CAN_MCR). This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments.

44.5.11.4 Pretended Networking Mode

This is a special low power mode used to receive wake up messages with low power consumption. This mode can be selected to operate together with Stop mode. Before entering in one of these low power modes, the PNET_EN bit in CAN_MCR Register must be asserted. Once in low power mode, CHI sub-block is shut down and CAN_PE sub-block is kept active, so that the Rx receive process is still active to filter incoming messages (see Receive Process under Pretended Networking Mode) as defined by the

configuration registers (see Pretended Networking Control 1 Register (CAN_CTRL1_PN)). Upon detecting a wake up event, a Wake Up interrupt is issued to the system.

To enter in Pretended Networking mode FlexCAN must be in normal mode (neither in Freeze, nor in Disable mode). Under Pretended Networking mode, FlexCAN keeps itself synchronized with the CAN BUS in Stop mode. Then, when Stop mode is requested, FlexCAN performs the following steps:

- Waits to be in Idle state, or else waits for the third bit of Intermission, and then checks it to be recessive.
- Sets the LPM_ACK bit in CAN_MCR Register.
- Requests the shutdown of the CHI sub-module clock, while keeping the PE sub-module clock active.

FlexCAN can exit Pretended Networking mode by the following ways:

- The CPU removing the Stop Mode request.
- FlexCAN will wait until Bus Idle or third bit of Intermission state to negate CAN_MCR[LPM_ACK] bit.

The above exit ways can be triggered either by the FlexCAN action upon detecting a wake up event and issuing the respective interrupt, or by the CPU itself upon being waked up by another mean. In consequence, FlexCAN will wait until the Bus Idle state or until the third bit of Intermission state to negate CAN_MCR[LPM_ACK] bit and resume to the Normal mode. This procedure ensures that FlexCAN will be synchronized to the CAN bus after exiting the Pretended Networking mode. The CPU must wait for the CAN_MCR[LPM_ACK] bit to be negated before performing any access to FlexCAN.

When PNET_EN bit in CAN_MCR is asserted, the CPU must disable the Self Wake Up feature by configuring SLF_WAK=0 in CAN_MCR register (see Module Configuration Register).

44.5.12 Interrupts

The module has many interrupt sources: interrupts due to message buffers and interrupts due to the ORed interrupts from MBs, Bus Off, Bus Off Done, Error, Error Fast (errors detected in the data phase of CAN FD format messages with the BRS bit set), Wake Up, Wake Up Match, Wake Up Timeout, Tx Warning, and Rx Warning.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception.

Each of the buffers has an assigned flag bit in the CAN_IFLAG registers. The bit is set when the corresponding buffer completes a successful transfer and is cleared when the CPU writes it to 1 (unless another interrupt is generated at the same time).

Note

It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Rx FIFO is enabled (CAN_MCR[RFEN] = 1) and DMA is disabled (CAN_MCR[DMA] = 0), the interrupts corresponding to MBs 0 to 7 have different meanings. Bit 24 of the CAN_IFLAG1 register becomes the "FIFO Overflow" flag; bit 25 becomes the "FIFO Warning" flag, bit 26 becomes the "Frames Available in FIFO" flag and bits 27-30 are unused. See the description of the Interrupt Flags 1 Register (CAN_IFLAG1) for more information.

If both Rx FIFO and DMA are enabled (CAN_MCR[RFEN] and CAN_MCR[DMA] = 1) the FlexCAN does not generate any FIFO interrupt. Bit 26 of the CAN_IFLAG1 register still indicates "Frames Available in FIFO" and generates a DMA request. Bits 24, 25, 27-30 are unused.

CAUTION

FIFO cannot be enabled when CAN FD feature is enabled.

For a combined interrupt where multiple MB interrupt sources are OR'd together, the interrupt is generated when any of the associated MBs (or FIFO, if applicable) generates an interrupt. In this case, the CPU must read the CAN_IFLAG registers to determine which MB or FIFO source caused the interrupt.

The interrupt sources for Bus Off, Bus Off Done, Error, Error Fast, Wake Up, Tx Warning and Rx Warning generate interrupts like the MB interrupt sources, and can be read from CAN_ESR1 register. The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in the CAN_CTRL1 Register; the Wake-Up interrupt mask bit is located in the CAN_MCR.

The interrupt sources for Pretended Networking (Wake up by Match Flag and Wake up by Timeout Flag) can be read in the CAN_WU_MTC Register and the respective interrupts masks bits are located in CAN_CTRL1_PN Register.

44.5.13 Bus interface

The CPU access to FlexCAN registers are subject to the following rules:

- Unrestricted read and write access to supervisor registers (registers identified with S/U in Table "Module Memory Map" in Supervisor Mode or with S only) results in access error.
- Read and write access to implemented reserved address space results in access error.
- Write access to positions whose bits are all currently read-only results in access error. If at least one of the bits is not read-only then no access error is issued. Write permission to positions or some of their bits can change depending on the mode of operation or transitory state. Refer to register and bit descriptions for details.
- Read and write access to unimplemented address space results in access error.
- Read and write access to RAM located positions during Low Power Mode results in access error.
- It is possible for the RXIMR memory region to be considered as general purpose memory and available for access. There are two ways of doing this:
  a. If CAN_MCR[IRMQ] is cleared, the individual masks (RXIMR) are disabled. In this case the RXIMR memory region is considered as general purpose memory.
  b. If CAN_MCR[MAXMB] is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Note that reserved words within RAM cannot be used. As an example, suppose FlexCAN's RAM can support up to 16 MBs, CAN_CTRL2[RFFN] is 0x0, and CAN_MCR[MAXMB] is programmed with zero. The maximum number of MBs in this case becomes one. The RAM starts at 0x0080, and the space from 0x0080 to 0x008F is used by the one MB. The memory space from 0x0090 to 0x017F is available. The space between 0x0180 and 0x087F is reserved. The space from 0x0880 to 0x0883 is used by the one Individual Mask and the available memory in the Mask Registers space would be from 0x0884 to 0x08BF. From 0x08C0 through 0x09DF there are reserved words for internal use which cannot be used as general purpose RAM. As a general rule, free memory space for general purpose depends only on MAXMB.

44.6 Initialization/application information

This section provide instructions for initializing the FlexCAN module.

44.6.1 FlexCAN initialization sequence

The FlexCAN module may be reset in three ways:

- Chip level hard reset, which resets all memory mapped registers asynchronously
- SOFTRST bit in MCR, which resets some of the memory mapped registers synchronously. See Table 44-4 to see what registers are affected by soft reset.

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The CAN_MCR[SOFTRST] bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in a low power mode. The low power mode should be exited and the clocks resumed before applying soft reset.

The clock source should be selected while the module is in Disable mode (see CAN_CTRL1[CLKSRC] bit). After the clock source is selected and the module is enabled (CAN_MCR[MDIS] bit negated), FlexCAN automatically goes to Freeze mode. In Freeze mode, FlexCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in CAN_MCR Register are set, the internal state machines are disabled and the FRZACK and NOTRDY bits in the CAN_MCR Register are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FlexCAN is put into Freeze mode (see Freeze mode). The following is a generic initialization sequence applicable to the FlexCAN module:

- Initialize the Module Configuration Register (CAN_MCR)
  - Enable the individual filtering per MB and reception queue features by setting the IRMQ bit
  - Enable the warning interrupts by setting the WRNEN bit
  - If required, disable frame self reception by setting the SRXDIS bit
  - Enable the Rx FIFO by setting the RFEN bit
  - If Rx FIFO is enabled and DMA is required, set DMA bit
  - If Pretended Networking mode is required, set PNET_EN bit
  - Enable the abort mechanism by setting the AEN bit
  - Enable the local priority feature by setting the LPRIOEN bit

- Initialize the Control 1 Register (CAN_CTRL1) and optionally the CAN Bit Timing Register (CAN_CBT). Initialize also the CAN FD CAN Bit Timing Register (CAN_FDCBT).
  - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW
  - Optionally determine the bit timing parameters: EPROPSEG, EPSEG1, EPSEG2, ERJW
  - Determine the CAN FD bit timing parameters: FPROPSEG, FPSEG1, FPSEG2, FRJW
  - Determine the bit rate by programming the PRESDIV field and optionally the EPRESDIV field
  - Determine the CAN FD bit rate by programming the FPRESDIV field
  - Determine the internal arbitration mode (LBUF bit)
- Initialize the Message Buffers
  - The Control and Status word of all Message Buffers must be initialized
  - If Rx FIFO was enabled, the ID filter table must be initialized
  - Other entries in each Message Buffer should be initialized as required
- Initialize the Rx Individual Mask Registers (CAN_RXIMRn)
- Set required interrupt mask bits in the CAN_IMASK Registers (for all MB interrupts), in CAN_MCR Register for Wake-Up interrupt and in CAN_CTRL1 / CAN_CTRL2 Registers (for Bus Off and Error interrupts)
- If Pretended Networking mode is enabled, configure the necessary registers for selective Wake Up
- Negate the HALT bit in CAN_MCR

After the last step listed above, FlexCAN attempts to synchronize to the CAN bus.

Chapter 45
Inter-Integrated Circuit (I2C)

45.1 Chip-specific I2C information

45.1.1 I2C configurations

45.1.1.1 I2C's stop-stop ack protocol

I^2C module provides the acknowledgement to the stop request after the current ongoing transaction are successfully completed.

45.2 Overview

This chapter describes the Inter-Integrated Circuit (I^2C) bus module implemented on this chip and presents the following topics:

- Introduction to I^2C
- External signal descriptions
- Memory map and register definition
- Functional description
- Initialization/application information

45.3 Introduction to I^2C

This section presents the following topics:

- Definition: I²C module
- Advantages of the I²C bus
- Module block diagram
- Features
- Modes of operation
- Definition: I²C conditions

45.3.1 Definition: I²C module

The I²C module is a functional unit that provides a two-wire— serial data (SDA) and serial clock (SCL) — bidirectional serial bus that provides a simple and efficient method of data exchange between this chip and other devices, such as microcontrollers, EEPROMs, real-time clock devices, analog-to-digital converters, and LCDs.

45.3.2 Advantages of the I²C bus

The synchronous, multiple-master two-wire I²C bus:

- Minimizes interconnections between devices
- Allows the connection of additional devices to the bus for expansion and system development
- Includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously
- Does not require an external address decoder

45.3.3 Module block diagram

The following figure shows a block diagram of the I²C module.

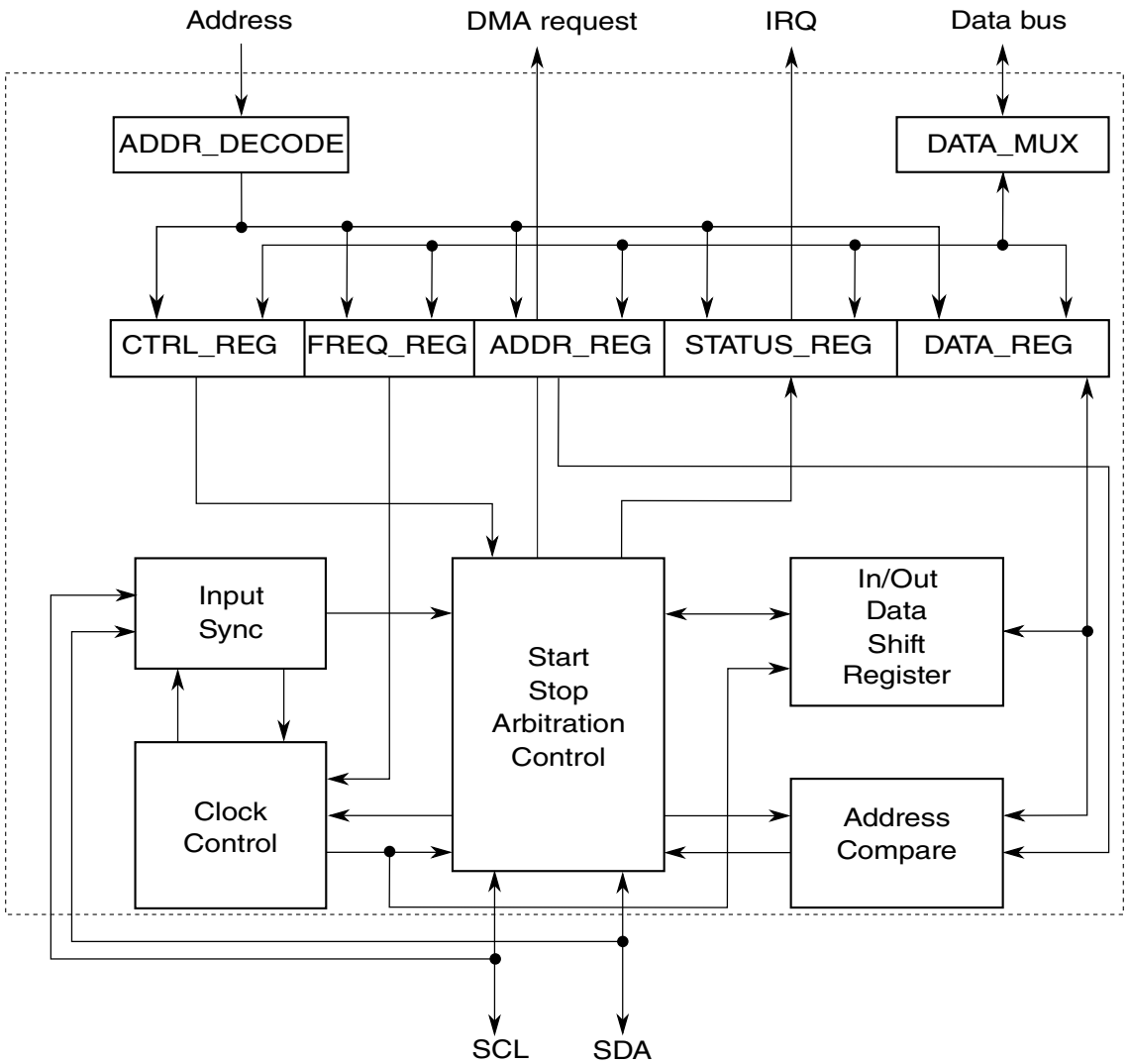


Figure 45-1. I²C block diagram

45.3.4 Features

The I²C module has the following key features:

- Compatible with I²C bus standard[1]
- Operating speeds
  - Up to 100 kbps in Standard Mode
  - Up to 400 kbps in Fast Mode

1. Compliant with I²C 2.0 standard with the exception that HS (high speed) mode is not supported

  - Operation at higher baud rates (up to a maximum of module clock/20) with reduced bus loading
  - Actual baud rate dependent on the SCL rise time (which depends on external pull-up resistor values and bus loading)
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- Basic DMA interface
- Maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF

45.3.5 Modes of operation

The I^2C module supports the chip modes described in the following table.

Table 45-1. Chip modes supported by the I^2C module

Chip mode	Description	Important notes
RUN	Basic mode of operation	—
STOP	The lowest-power mode that allows the chip to turn off all the clocks to the I^2C module	The I^2C module can enter this mode when there are no active transfers (active data between valid START and STOP conditions) on the bus. See STOP mode.
DEBUG	Allows the chip to freeze all ongoing activities (such as an ongoing transaction, counter values, and register status) for debugging	See DEBUG mode.

In addition to chip modes, the I^{2}C module has several module-specific modes. These are described in the following table.

Table 45-2. Module-specific modes supported by the I^{2}C module

Module mode	Description	Important notes
Master mode	The I^{2}C module is the driver of the SDA line.	• Do not use the I^{2}C module's slave address as a calling address. • The I^{2}C module cannot be a master and a slave simultaneously.
Slave mode	The I^{2}C module is not the driver of the SDA line.	• Enable the I^{2}C module before a START condition from a non-I^{2}C master is detected. • By default the I^{2}C module performs as a slave receiver.

45.3.6 Definition: I^{2}C conditions

The following table shows the I^{2}C-specific conditions defined for the I^{2}C module.

Table 45-3. I^{2}C Conditions

Condition	Description
START	A condition that denotes the beginning of a new data transfer and awakens all slaves. Each data transfer contains several bytes of data. It is defined as a high-to-low transition of SDA while SCL is high, as shown in the following figure.
STOP	A condition generated by the master to terminate a transfer and free the bus. It is defined as a low-to-high transition of SDA while SCL is high, as shown in the following figure.
Repeated START	A START condition that is generated without a STOP condition to terminate the previous transfer.

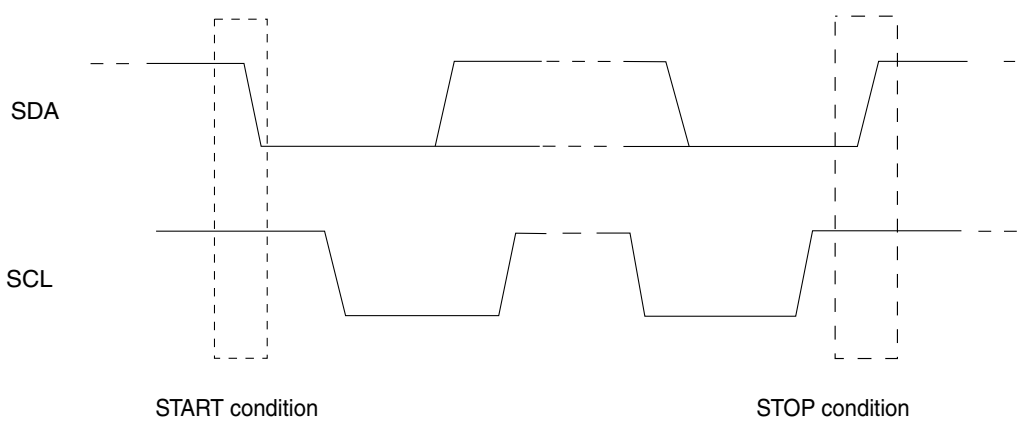


Figure 45-2. START and STOP conditions

45.4 External signal descriptions

This section presents the following topics:

- Signal overview
- Detailed external signal descriptions

45.4.1 Signal overview

The I^2C module uses the Serial Data (SDA) and Serial Clock (SCL) signals as a communication interconnect with other devices. The signal patterns driven on the SDA signal represent address, data, or read/write information at different stages of the protocol.

All devices connected to the SDA and SCL signals must have open-drain or open-collector outputs. The logical AND function is performed on both signals with external pull-up resistors. For the electrical characteristics of these signals, see the data sheet for this chip.

45.4.2 Detailed external signal descriptions

The SDA and SCL signals are described in the following table.

Table 45-4. External signal descriptions

Signal	Description
SCL	Bidirectional serial clock line of the module, compatible with the I^2C bus specification
SDA	Bidirectional serial data line of the module, compatible with the I^2C bus specification

45.5 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the I^2C module. It presents the following topics:

- Register accessibility
- Register figure conventions

- I2C Bus Address Register (I2C_IBAD)
- I2C Bus Frequency Divider Register (I2C_IBFD)
- I2C Bus Control Register (I2C_IBCR)
- I2C Bus Status Register (I2C_IBSR)
- I2C Bus Data I/O Register (I2C_IBDR)
- I2C Bus Interrupt Config Register (I2C_IBIC)
- I2C Bus Debug Register (I2C_IBDBG)

45.5.1 Register accessibility

Address location 0x0007 is a reserved location, but access to this location will not generate any bus error.

All registers are accessible via 8-bit, 16-bit, or 32-bit accesses. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries.

As an example, the IBFD is at address offset 0x01, and can be accessed in any of the following ways:

- 8-bit R/W access of address offset 0x0001
- Second byte of 16-bit R/W access of address offset 0x0000
- Second byte of 32-bit R/W access of address offset 0x0000

The IBFD register cannot be accessed by a 16- or 32-bit RW operation at address offset 0x0001 because those operations require an address aligned to a 16- or 32-bit boundary.

45.5.2 Register figure conventions

The register figures show the field structure using the conventions in the following figure.

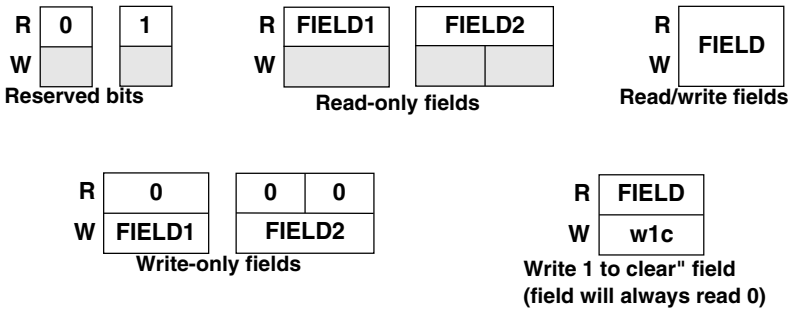


Figure 45-3. Register figure convention

The memory map for the I^{2}C module is given below. The total address for each register is the sum of the base address for the I^{2}C module and the address offset for each register.

I2C memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	I2C Bus Address Register (I2C_IBAD)	8	R/W	00h	45.5.3/1684
1	I2C Bus Frequency Divider Register (I2C_IBFD)	8	R/W	00h	45.5.4/1685
2	I2C Bus Control Register (I2C_IBCR)	8	R/W	80h	45.5.5/1685
3	I2C Bus Status Register (I2C_IBSR)	8	R/W	80h	45.5.6/1687
4	I2C Bus Data I/O Register (I2C_IBDR)	8	R/W	00h	45.5.7/1688
5	I2C Bus Interrupt Config Register (I2C_IBIC)	8	R/W	00h	45.5.8/1689
6	I2C Bus Debug Register (I2C_IBDBG)	8	R/W	00h	45.5.9/1690

45.5.3 I2C Bus Address Register (I2C_IBAD)

This register contains the address the I^{2}C Bus will respond to when addressed as a slave. This is not the address sent on the bus during the address transfer.

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7
Read	ADR							0
Write								
Reset	0	0	0	0	0	0	0	0

I2C_IBAD field descriptions

Field	Description
0–6 ADR	Slave Address. Specific slave address to be used by the I^{2}C Bus module.

Table continues on the next page...

I2C_IBAD field descriptions (continued)

Field	Description
	NOTE: The default mode of I^2C Bus is slave mode for an address match on the bus.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.5.4 I2C Bus Frequency Divider Register (I2C_IBFD)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 1h offset = 1h

Bit	0	1	2	3	4	5	6	7
Read Write	IBC							
Reset	0	0	0	0	0	0	0	0

I2C_IBFD field descriptions

Field	Description
0–7 IBC	I-Bus Clock Rate. This field is used to prescale the bus clock for bit rate selection. See Clock rate and IBFD settings.

45.5.5 I2C Bus Control Register (I2C_IBCR)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7
Read	MDIS	IBIE	MSSL	TXRX	NOACK	0	DMAEN	Reserved
Write						RSTA		
Reset	1	0	0	0	0	0	0	0

I2C_IBCR field descriptions

Field	Description
0 MDIS	Module disable. This bit controls the software reset of the entire I^2C Bus module. **NOTE:** If the I^2C Bus module is enabled in the middle of a byte transfer, the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent

Table continues on the next page...

I2C_IBCR field descriptions (continued)

Field	Description
	start condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the I²C Bus module losing arbitration, after which, bus operation would return to normal. 0 The I²C Bus module is enabled. This bit must be cleared before any other IBCR bits have any effect 1 The module is reset and disabled. This is the power-on reset situation. When high, the interface is held in reset, but registers can still be accessed. Status register bits (IBSR) are not valid when module is disabled.
1 IBIE	I-Bus Interrupt Enable. 0 Interrupts from the I²C Bus module are disabled. This does not clear any currently pending interrupt condition. 1 Interrupts from the I²C Bus module are enabled. An I²C Bus interrupt occurs provided the IBIF bit in the status register is also set.
2 MSSL	Master/Slave mode select. When this bit is changed from 0 to 1, a START signal is generated on the bus and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave. A STOP signal should be generated only if the IBIF flag is set. This field is cleared without generating a STOP signal when the master loses arbitration. 0 Slave Mode 1 Master Mode
3 TXRX	Transmit/Receive mode select. This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. 0 Receive 1 Transmit
4 NOACK	Data Acknowledge disable. This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The I²C module will always acknowledge address matches, provided it is enabled, regardless of the value of NOACK. **NOTE:** Values written to this bit are only used when the I²C Bus is a receiver, not a transmitter. 0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte of data 1 No acknowledge signal response is sent (i.e., acknowledge bit = 1)
5 RSTA	Repeat Start. Writing a one to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration. 0 No effect 1 Generate repeat start cycle
6 DMAEN	DMA Enable. When this bit is set, the DMA Tx and Rx lines will be asserted when the I²C module requires data to be read or written to the data register. No Transfer Done interrupts will be generated when this bit is set, however an interrupt will be generated if the loss of arbitration or addressed as slave conditions occur. The DMA mode is only valid when the I²C module is configured as a Master and the DMA transfer still requires CPU intervention at the start and the end of each frame of data. See the DMA Application Information section for more details. 0 Disable the DMA TX/RX request signals 1 Enable the DMA TX/RX request signals

Table continues on the next page...

I2C_IBCR field descriptions (continued)

Field	Description
7 Reserved	This field is reserved. Although this bit supports read/write access, writing to this bit is not recommended, and can produce unexpected results.

45.5.6 I2C Bus Status Register (I2C_IBSR)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 3h offset = 3h

Bit	0	1	2	3	4	5	6	7
Read	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
Write				w1c			w1c	
Reset	1	0	0	0	0	0	0	0

I2C_IBSR field descriptions

Field	Description
0 TCF	Transfer complete. While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. **NOTE:** This bit is only valid during or immediately following a transfer to the I^2C module or from the I^2C module. 0 Transfer in progress 1 Transfer complete
1 IAAS	Addressed as a slave. When its own specific address (I-Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the IBIE is set. Then the CPU needs to check the SRW bit and set the TXRX field accordingly. Writing to the I-Bus Control Register clears this bit. 0 Not addressed 1 Addressed as a slave
2 IBB	Bus busy. This bit indicates the status of the bus. When a START signal is detected, IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state. **NOTE:** Software must ensure that the I^2C bus is idle by checking the IBSR[IBB] field (bus busy) before switching to master mode and attempting a START cycle. 0 Bus is Idle 1 Bus is busy

Table continues on the next page...

I2C_IBSR field descriptions (continued)

Field	Description
3 IBAL	Arbitration Lost. The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances: • SDA is sampled low when the master drives a high during an address or data transmit cycle. • SDA is sampled low when the master drives a high during the acknowledge bit of a data receive cycle. • A start cycle is attempted when the bus is busy. • A repeated start cycle is requested in slave mode. • A stop condition is detected when the master did not request it. This bit must be cleared by software, by writing a one to it. A write of zero has no effect.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SRW	Slave Read/Write. When the IAAS bit is set, this bit indicates the value of the R/W command bit of the calling address sent from the master. This bit is only valid when the I-Bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated. By reading this field, the CPU can detect slave transmit/receive mode according to the command of the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
6 IBIF	I-Bus Interrupt Flag. The IBIF bit is set when one of the following conditions occurs: • Arbitration lost (IBAL bit set) • Byte transfer complete (TCF bit set and DMAEN bit not set) • Addressed as slave (IAAS bit set) • NoAck from Slave (MS & Tx bits set) • I^2C Bus going idle (IBB high-low transition and enabled by BIIE) A processor interrupt request will be caused if the IBIE bit is set. This bit must be cleared by software, by writing a one to it. A write of zero has no effect on this bit. In DMA mode (DMAEN set) a byte transfer complete condition will not trigger the setting of IBIF. All other conditions still apply.
7 RXAK	Received Acknowledge. This is the value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock. This bit is valid only after transfer is complete. 0 Acknowledge received 1 No acknowledge received

45.5.7 I2C Bus Data I/O Register (I2C_IBDR)

In master transmit mode, when data is written to the IBDR, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred.

NOTE

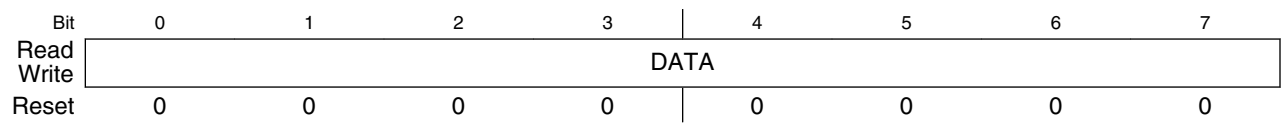
The IBCR[TXRX] field must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the I^2C is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the most recent byte received while the I^2C is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the I^2C bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to IBDR following assertion of MSSL is used for the address transfer and should comprise the calling address (in position DATA[7:1]) concatenated with the required R/ $\overline{W}$ bit (in position D0).

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 4h offset = 4h



I2C_IBDR field descriptions

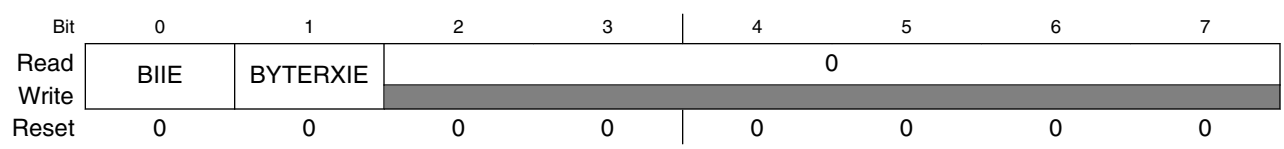
Field	Description
0–7 DATA	Data transmitted or received

45.5.8 I2C Bus Interrupt Config Register (I2C_IBIC)

To program BIIE = 1, you must ensure that IBCR[MDIS] = 0.

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 5h offset = 5h



I2C_IBIC field descriptions

Field	Description
0 BIIE	Bus Idle Interrupt Enable bit. This config bit can be used to enable the generation of an interrupt once the I^2C bus becomes idle. Once this bit is set, an IBB high-low transition will set the IBIF bit. This feature can be used to signal to the CPU the completion of a STOP on the I^2C bus. 0 Bus Idle Interrupts disabled 1 Bus Idle Interrupts enabled
1 BYTERXIE	Byte receive interrupt enable This field is used to generate an interrupt every time the I^2C master/slave receives a new byte. This feature can be useful when an I^2C master is receiving data from a slave that is transmitting on an irregular basis. BYTERXIE is updated only when the I^2C is enabled (IBCR[MDIS]=0).
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

45.5.9 I2C Bus Debug Register (I2C_IBDBG)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: 0h base + 6h offset = 6h

Bit	0	1	2	3
Read	0			
Write				
Reset	0	0	0	0

Bit	4	5	6	7
Read	0	BYTE_RX	IPG_DEBUG_HALTED	IPG_DEBUG_EN
Write		w1c		
Reset	0	0	0	0

I2C_IBDBG field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 BYTE_RX	This field gets set when an I^2C master/slave receives a new byte. An interrupt will be generated if IBIC[BYTERXIE] = 1.
6 IPG_DEBUG_ HALTED	Debug Halted Bit: This is s status bit which can be read back after asserting the debug enable signal so as to know if the IP has entered the debug mode or not. 0 IP is still executing a transaction 1 IP has entered the debug mode

Table continues on the next page...

I2C_IBDBG field descriptions (continued)

Field	Description
7 IPG_DEBUG_EN	Debug enable bit. This bit is used to enable IP enter the debug mode provided the IPG DEBUG signal is high. All the registers, counter values and status bits are frozen and can be accessed by the CPU. **NOTE:** If the assertion of this bit along with the IPG DEBUG signal happens in the middle of a transaction, IP enters the debug mode only after successful completion of the current transaction after which no further transaction can take place until the debug mode is exited. 0 Normal operation, Bus Idle Interrupts disabled 1 IP is in debug mode

45.6 Functional description

This section presents the following topics:

- Notes about module operation
- Transactions
- Arbitration procedure
- Clock behavior
- Interrupts
- DEBUG mode
- DMA interface

45.6.1 Notes about module operation

- The I^2C module always performs as a slave receiver by default, unless explicitly programmed to be a master or slave transmitter.
- When the I^2C module is acting as a master, it must not try to call its own slave address.

45.6.2 Transactions

This section presents the following topics:

- Protocol overview

- Transaction protocol definitions
- High-level protocol steps
- START condition
- Slave address transmission
- Data transmission
- STOP condition
- Repeated START condition

45.6.2.1 Protocol overview

The following figure shows the behavior of SCL and SDA during a typical I^2C transaction.

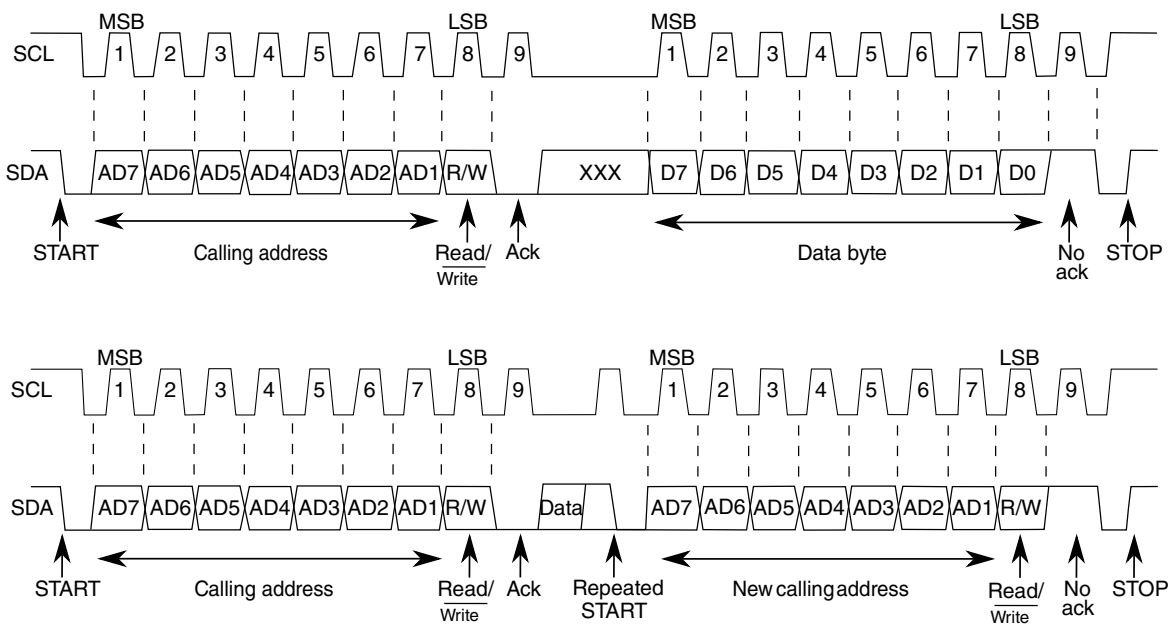


Figure 45-4. I^2C transaction protocol

45.6.2.2 Transaction protocol definitions

This section defines several important terms presented in Figure 45-4.

Table 45-5. I²C definitions

Term	Definition
START	A START condition, as defined in Section 1.2.6, Definition: I²C conditions"
STOP	A STOP condition, as defined in Section 1.2.6, Definition: I²C conditions"
Calling (slave) address	A seven-bit address used to identify a slave on the I²C bus. The requirements for specifying this address are presented in Section 1.5.2.3, I²C calling address requirements."
Read/write (R/W)	A bit that specifies the direction of the data transfer to the slave as follows: • 0=The data is being transferred from the master to the slave ("write") • 1=The data is being transferred from the slave to the master ("read")
Ack	A bit that specifies the acknowledgement of a calling address, indicated by pulling SDA low.

45.6.2.3 I²C calling address requirements

The calling addresses of the devices used on an I²C network are subject to the following requirements:

- Each slave must have a unique calling address.
- A master must not transmit a calling address that is the same as its own slave address.

45.6.2.4 High-level protocol steps

The I²C protocol conceptually supports two types of transfers, which are illustrated in Figure 45-4. The significant steps in these transfers are presented in the following table. Details of each of these steps are presented in subsequent sections.

Table 45-6. I²C high-level protocol steps

Standard Transfer	Repeated START Transfer
1. START condition	1. START condition
2. Slave target or general call address transmission	2. Slave target or general call address transmission
3. Acknowledgment from slave	3. Acknowledgment from slave
4. Data transfer	4. Data transfer
5. STOP condition	5. Repeated START condition
6. (repeat Steps 1–4)	6. (repeat Steps 2–4 as needed)
	7. STOP condition.
	8. (repeat Steps 1–7)

45.6.2.5 START condition

When the bus is free, that is, no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START condition (see Definition: I^{2}C conditions). This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

See Clock rate and IBFD settings and I2C Bus Frequency Divider Register (I2C_IBFD) for the associated timing requirements.

45.6.2.6 Slave address transmission

The master transmits the slave address on the next clock cycle after the START condition (see START condition). The process of slave address transmission is presented in the following table.

Table 45-7. Slave address transmission process

Step	Action
1	The master transmits the seven-bit slave address.
2	The master transmits the R/W bit.
3	Each slave examines the transmitted address and compares it to its own. If the addresses match, the slave device returns the acknowledge bit on the ninth SCL clock cycle.
4	The master waits for the acknowledge bit and determines the next step as follows: • The acknowledge bit is set: The master must initiate a data transfer followed by either a STOP condition or a repeated START condition. • The acknowledge bit is cleared: The master must wait for SCL to return to logic zero.

45.6.2.7 Data transmission

A data transfer session has the following characteristics:

- Can transmit one or more bytes of data
- Awakens all slaves
- Proceeds on a byte-by-byte basis in the direction specified by the R/W bit sent by the calling master

The transmitted data is subject to the following requirements:

- Each data byte must consist of 8 bits.
- Data bits can be changed only while SCL is low and must be held stable while SCL is high.
- One data bit is transmitted during one SCL clock pulse.
- The most significant bit (msb) must be transmitted first.
- Each data byte must be followed by an acknowledge bit on the ninth SCL clock pulse.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA line for the master to generate a STOP or a START condition.

45.6.2.8 STOP condition

The master can terminate the communication by generating a STOP condition (see Definition: I^2C conditions). It can do so even if the slave has generated an acknowledge, at which point the slave must release the bus.

A master is not required to send a STOP condition at the end of every transfer. For more information, see Repeated START condition.

See Clock rate and IBFD settings and I2C Bus Frequency Divider Register (I2C_IBFD) for the associated timing requirements.

45.6.2.9 Repeated START condition

The I^2C protocol also supports a repeated START condition, which can be generated without a preceding STOP condition. A master device can use this condition to communicate with another slave or with the same slave in a different mode without releasing the bus. This condition is illustrated in the second timing diagram of Figure 45-4.

45.6.3 Arbitration procedure

The I^2C bus is a true multi-master bus that allows more than one master to be connected to it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure. A bus master loses arbitration if it transmits logic "1" while another master transmits logic "0". The losing masters immediately switch over to slave mode and stop driving the SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

45.6.4 Clock behavior

This section presents the following topics:

- Clock synchronization
- Clock stretching
- Handshaking
- Clock rate and IBFD settings

45.6.4.1 Clock synchronization

Due to the wired-AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the master drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached.

However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, the synchronized clock signal, SCL, is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see the following figure). When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the devices' clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

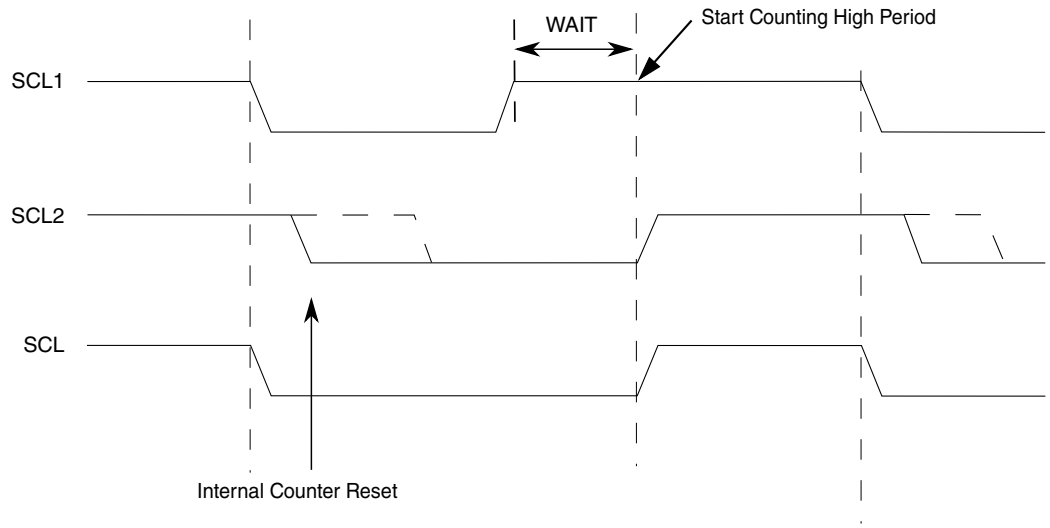


Figure 45-5. I²C bus clock synchronization

45.6.4.2 Clock stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

45.6.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait state until the slave releases the SCL line.

45.6.4.4 Clock rate and IBFD settings

45.6.4.4.1 Timing definitions

Table 45-8. Timing definitions relevant to clock rate and IBFD settings

Term	Definition
SCL Divider	The factor used to prescale the CPU clock for bit rate selection (see Figure 45-6 and Table 45-9)
SCL period	(CPU clock period) × (SCL Divider)

Table continues on the next page...

Table 45-8. Timing definitions relevant to clock rate and IBFD settings (continued)

Term	Definition
SCL Hold	The required number of CPU clocks to generate a START or STOP condition (see Figure 45-6 and Table 45-9)
SDA Hold	See Figure 45-7 and Table 45-9

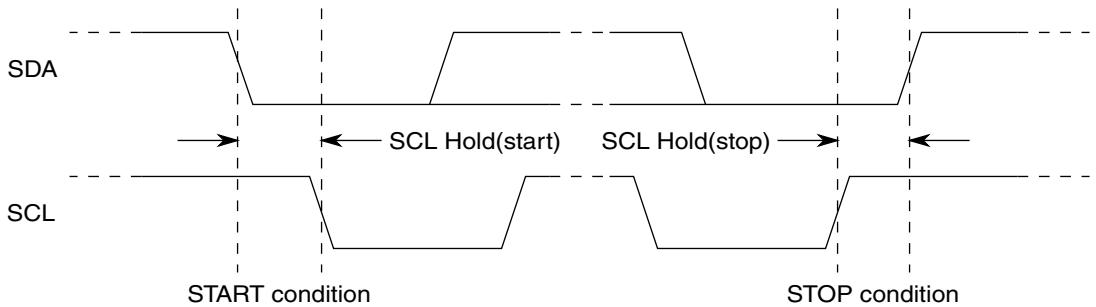


Figure 45-6. SCL Divider and SDA Hold

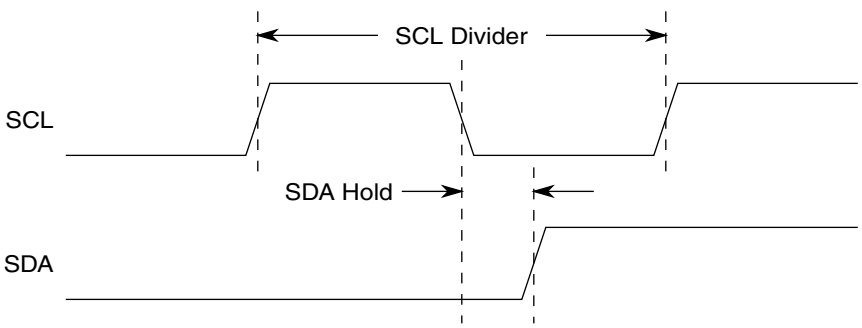


Figure 45-7. SDA Hold time

45.6.4.4.2 Divider and hold values

Table 45-9. I²C divider and hold values

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL=1	00	20	7	6	11
	01	22	7	7	12
	02	24	8	8	13
	03	26	8	9	14
	04	28	9	10	15
	05	30	9	11	16
	06	34	10	13	18
	07	40	10	16	21
	08	28	7	10	15
	09	32	7	12	17

Table continues on the next page...

Table 45-9. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	0A	36	9	14	19
	0B	40	9	16	21
	0C	44	11	18	23
	0D	48	11	20	25
	0E	56	13	24	29
	0F	68	13	30	35
	10	48	9	18	25
	11	56	9	22	29
	12	64	13	26	33
	13	72	13	30	37
	14	80	17	34	41
	15	88	17	38	45
	16	104	21	46	53
	17	128	21	58	65
	18	80	9	38	41
	19	96	9	46	49
	1A	112	17	54	57
	1B	128	17	62	65
	1C	144	25	70	73
	1D	160	25	78	81
	1E	192	33	94	97
	1F	240	33	118	121
	20	160	17	78	81
	21	192	17	94	97
	22	224	33	110	113
	23	256	33	126	129
MUL=1	24	288	49	142	145
	25	320	49	158	161
	26	384	65	190	193
	27	480	65	238	241
	28	320	33	158	161
	29	384	33	190	193
	2A	448	65	222	225
	2B	512	65	254	257
	2C	576	97	286	289
	2D	640	97	318	321
	2E	768	129	382	385
	2F	960	129	478	481

Table continues on the next page...

Table 45-9. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	30	640	65	318	321
	31	768	65	382	385
	32	896	129	446	449
	33	1024	129	510	513
	34	1152	193	574	577
	35	1280	193	638	641
	36	1536	257	766	769
	37	1920	257	958	961
	38	1280	129	638	641
	39	1536	129	766	769
	3A	1792	257	894	897
	3B	2048	257	1022	1025
	3C	2304	385	1150	1153
	3D	2560	385	1278	1281
	3E	3072	513	1534	1537
	3F	3840	513	1918	1921
MUL=2	40	40	14	12	22
	41	44	14	14	24
	42	48	16	16	26
	43	52	16	18	28
	44	56	18	20	30
	45	60	18	22	32
	46	68	20	26	36
	47	80	20	32	42
	48	56	14	20	30
	49	64	14	24	34
MUL=2	4A	72	18	28	38
	4B	80	18	32	42
	4C	88	22	36	46
	4D	96	22	40	50
	4E	112	26	48	58
	4F	136	26	60	70
	50	96	18	36	50
	51	112	18	44	58
	52	128	26	52	66
	53	144	26	60	74
	54	160	34	68	82
	55	176	34	76	90

Table continues on the next page...

Table 45-9. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	56	208	42	92	106
	57	256	42	116	130
	58	160	18	76	82
	59	192	18	92	98
	5A	224	34	108	114
	5B	256	34	124	130
	5C	288	50	140	146
	5D	320	50	156	162
	5E	384	66	188	194
	5F	480	66	236	242
	60	320	34	156	162
	61	384	34	188	194
	62	448	66	220	226
	63	512	66	252	258
	64	576	98	284	290
	65	640	98	316	322
	66	768	130	380	386
	67	960	130	476	482
	68	640	66	316	322
	69	768	66	380	386
	6A	896	130	444	450
	6B	1024	130	508	514
MUL=2	6C	1152	194	572	578
	6D	1280	194	636	642
	6E	1536	258	764	770
	6F	1920	258	956	962
	70	1280	130	636	642
	71	1536	130	764	770
	72	1792	258	892	898
	73	2048	258	1020	1026
	74	2304	386	1148	1154
	75	2560	386	1276	1282
	76	3072	514	1532	1538
	77	3840	514	1916	1922
	78	2560	258	1276	1282
	79	3072	258	1532	1538
	7A	3584	514	1788	1794
	7B	4096	514	2044	2050

Table continues on the next page...

Table 45-9. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	7C	4608	770	2300	2306
	7D	5120	770	2556	2562
	7E	6144	1026	3068	3074
	7F	7680	1026	3836	3842
MUL=4	80	80	28	24	44
	81	88	28	28	48
	82	96	32	32	52
	83	104	32	36	56
	84	112	36	40	60
	85	120	36	44	64
	86	136	40	52	72
	87	160	40	64	84
	88	112	28	40	60
	89	128	28	48	68
	8A	144	36	56	76
	8B	160	36	64	84
	8C	176	44	72	92
	8D	192	44	80	100
	8E	224	52	96	116
	8F	272	52	120	140
	90	192	36	72	100
	91	224	36	88	116
	92	256	52	104	132
MUL=4	93	288	52	120	148
	94	320	68	136	164
	95	352	68	152	180
	96	416	84	184	212
	97	512	84	232	260
	98	320	36	152	164
	99	384	36	184	196
	9A	448	68	216	228
	9B	512	68	248	260
	9C	576	100	280	292
	9D	640	100	312	324
	9E	768	132	376	388
	9F	960	132	472	484
	A0	640	68	312	324
	A1	768	68	376	388

Table continues on the next page...

Table 45-9. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	A2	896	132	440	452
	A3	1024	132	504	516
	A4	1152	196	568	580
	A5	1280	196	632	644
	A6	1536	260	760	772
	A7	1920	260	952	964
	A8	1280	132	632	644
	A9	1536	132	760	772
	AA	1792	260	888	900
	AB	2048	260	1016	1028
	AC	2304	388	1144	1156
	AD	2560	388	1272	1284
	AE	3072	516	1528	1540
	AF	3840	516	1912	1924
	30	2560	260	1272	1284
	B1	3072	260	1528	1540
	B2	3584	516	1784	1796
	B3	4096	516	2040	2052
	B4	4608	772	2296	2308
	B5	5120	772	2552	2564
	B6	6144	1028	3064	3076
	B7	7680	1028	3832	3844
	B8	5120	516	2552	2564
MUL=4	B9	6144	516	3064	3076
	BA	7168	1028	3576	3588
	BB	8192	1028	4088	4100
	BC	9216	1540	4600	4612
	BD	10240	1540	5112	5124
	BE	12288	2052	6136	6148
	BF	15360	2052	7672	7684

45.6.5 Interrupts

This section presents the following topics:

- Interrupt vector
- Interrupt description

45.6.5.1 Interrupt vector

The I^{2}C module uses only one interrupt vector.

Table 45-10. Interrupt summary

Interrupt	Offset	Vector	Priority	Source	Description
I^{2}C Interrupt	—	—	—	IBAL, TCF, IAAS, IBB bits in IBSR register	When any of IBAL, TCF or IAAS bits is set, an interrupt may be caused based on Arbitration lost, Transfer Complete or Address Detect conditions. If enabled by BIIE, the de-assertion of IBB can also cause an interrupt, indicating that the bus is idle.

45.6.5.2 Interrupt description

There are five types of internal interrupts in the I^{2}C. The interrupt service routine can determine the interrupt type by reading the Status register.

I^{2}C Interrupt can be generated on the following events:

- Arbitration Lost condition (IBAL bit set)
- Byte Transfer condition (TCF bit set and DMAEN bit not set)
- Address Detect condition (IAAS bit set)
- No Acknowledge from slave received when expected
- Bus Going Idle (IBB bit not set)

The I^{2}C interrupt is enabled by the IBCR[IBIE] bit. It must be cleared by writing '1' to the IBIF bit in the interrupt service routine. The Bus Going Idle interrupt needs to be additionally enabled by the IBIC[BIIE] bit.

45.6.6 STOP mode

This mode allows the software to put the I^{2}C module in power-down state. Once the STOP request is asserted, the I^{2}C module comes to a graceful halt after completing all the ongoing transactions.

As soon as the I^{2}C module enters STOP mode:

- The I^{2}C clock is disabled.
- No transaction can take place.
- All registers are inaccessible.

The I^{2}C module enters this mode only after successfully completing the current ongoing transaction, hence the low-power request is acknowledged once the STOP condition occurs. The user must ensure that the bus is free when STOP is requested. To request STOP for ongoing transmission the user must wait until the transmission is complete, followed by clearing of the IBCR[TXRX] field. See the figure below for more details.

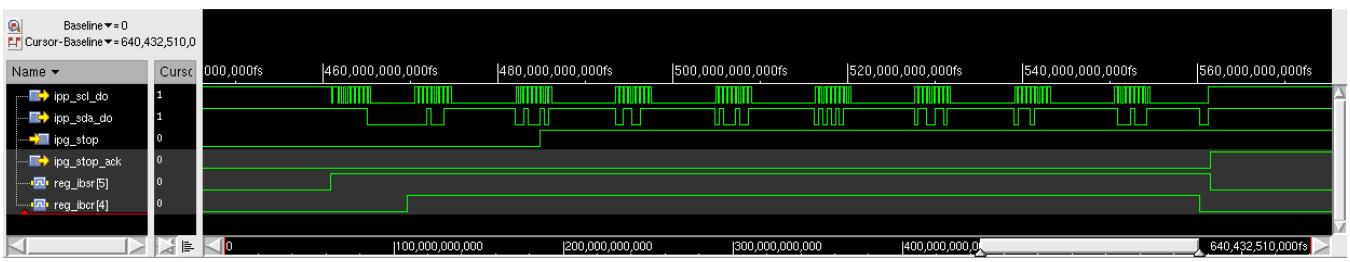


Figure 45-8. I^{2}C stop mode behavior when master is receiving and slave is transmitting

45.6.7 DEBUG mode

This mode allows CPU to debug the I^{2}C by freezing all the counters, registers and status bits. Once the Debug request is asserted along with IBDBG[IPG_DEBUG_EN] bit, I^{2}C comes to a graceful halt after completing all the ongoing transactions. A Debug halted signal IBDBG[IPG_DEBUG_HALTED] is also asserted to indicate that the debug request has been successfully serviced and is de-asserted when the Debug request is de-asserted.

As soon as the I^{2}C module enters DEBUG mode:

- No transaction can take place.
- All the registers, counters and status bits are frozen. They all can be accessed by the CPU to enable the debugging.

The I²C module enters this mode only after successfully completing the current ongoing transaction. For example, if the current transaction consists of 8 bytes and the debug mode was initiated by user at the time of the second byte, the IPG Debug Halted signal will be asserted after the 8th byte is transmitted/received. See the simulation result in Figure 45-9 for more details.

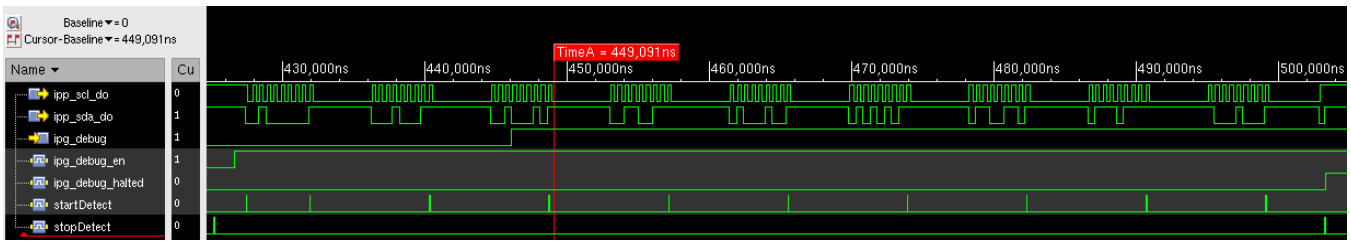


Figure 45-9. Simulation result of IPG Debug Halted in case of frame transmission

If any DMA transaction (transmit or receive) is in progress, and if the debug mode is requested while a byte is being transmitted or received, the I²C module enters DEBUG mode after completing the transmission or reception of the current byte. As soon as the module exits DEBUG mode, transmission or reception of the remaining bytes resumes. Please see the simulation snapshot shown in Figure 45-10 for details where the I²C is transmitting 8 bytes using DMA and DEBUG mode is requested while a second byte is being transmitted. DEBUG mode is entered after successfully transmitting the second byte (IPG Debug Halted signal asserted). As soon as the module exits DEBUG mode (IPG Debug Halted signal de-asserted), transmission resumes successfully.

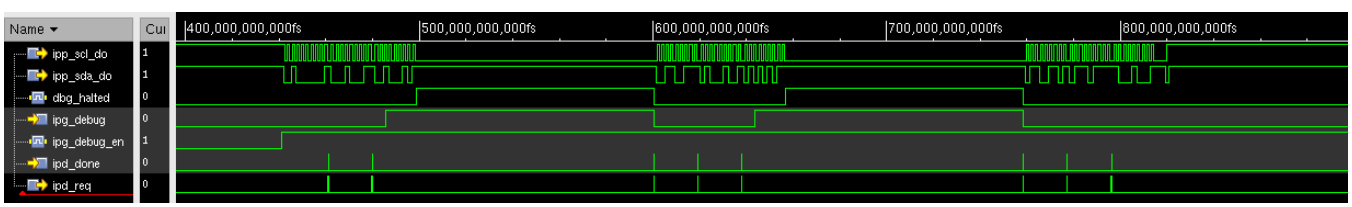


Figure 45-10. Simulation result of IPG Debug Halted in case of frame transmission using DMA

No more transaction can take place until the debug signal is de-asserted after which the I²C module starts functioning normally. There is a status halted signal IBDBG[IPG_DEBUG_HALTED] to indicate to the user that the I²C module has entered the DEBUG mode. See the following figure for more details.

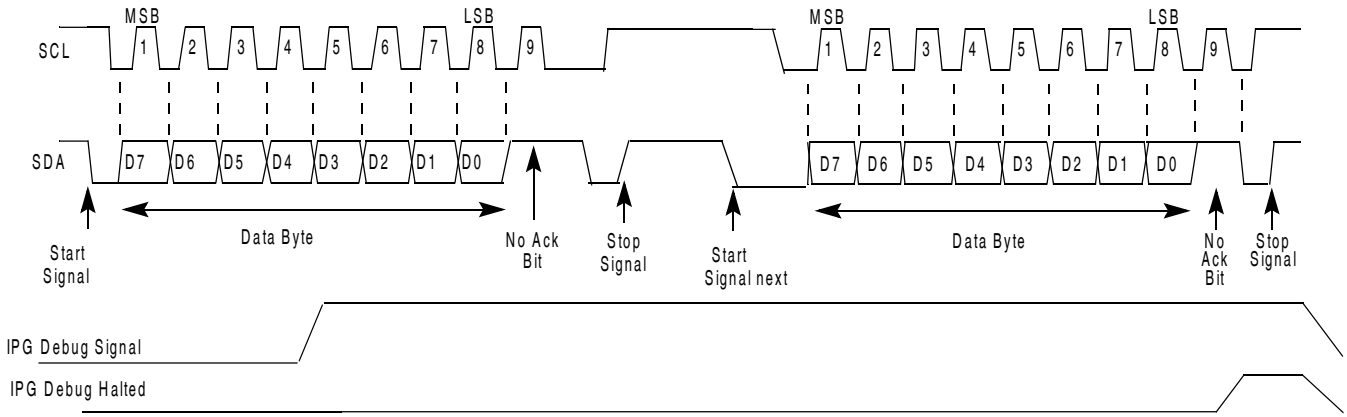


Figure 45-11. DEBUG mode

The following figure shows a case of DEBUG mode with repeated START transaction. In this case, if the debug signal is asserted in between the transaction, the entire transaction with multiple repeated start will be completed before the I^2C module enters DEBUG mode.

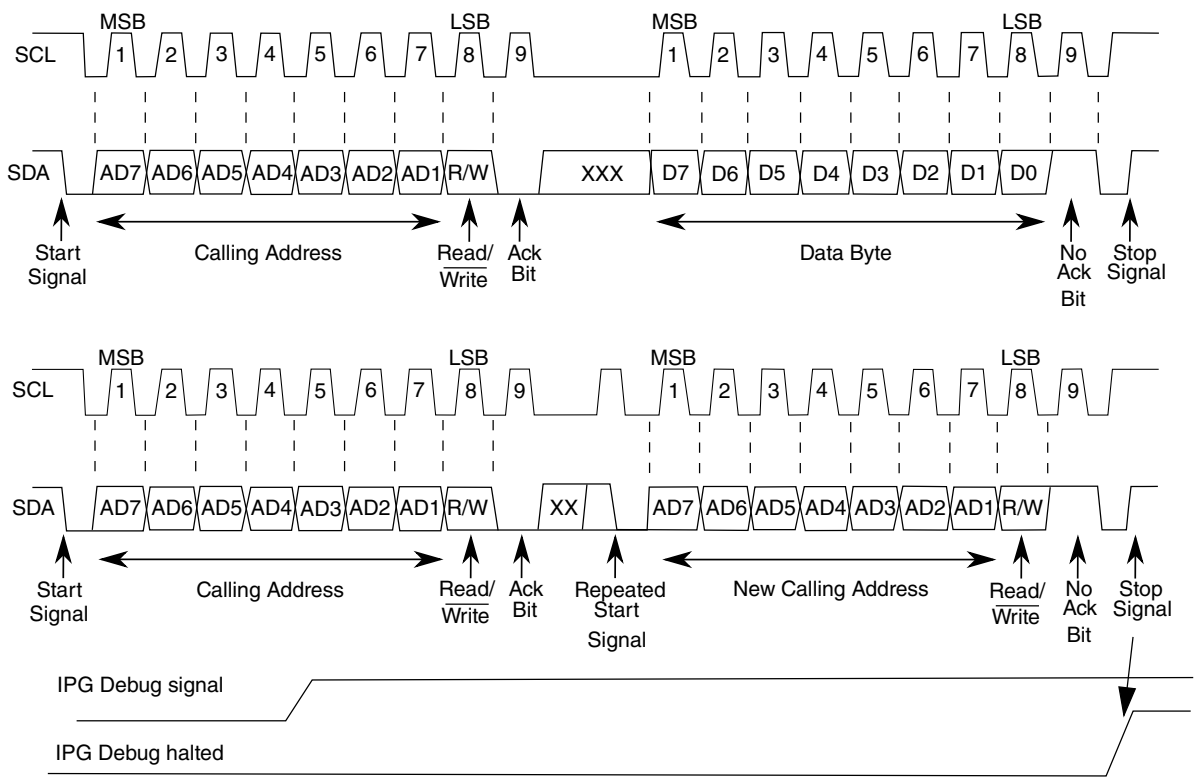


Figure 45-12. DEBUG mode with repeated start

45.6.8 DMA interface

A simple DMA interface is implemented so that the I^2C can request data transfers with minimal support from the CPU (see DMA application information). DMA mode is enabled by setting bit 1 in the Control Register (IBCR).

The DMA interface is operational when the I^2C module is configured for Master mode.

At least three bytes of data per frame must be transferred from/to the slave when using DMA mode, although in practice it will only be worthwhile using the DMA mode when there is a large number of data bytes to transfer per frame.

Two internal signals, TX request and RX request, are used to signal the DMA controller when the I^2C module requires data to be written or read from the data register.

Further details of the DMA interface can be found in the Initialization/application information, of this document.

45.7 Initialization/application information

This section presents the following topics:

- Recommended interrupt service flow
- General programming guidelines (for both master and slave mode)
- Programming guidelines specific to master mode
- Programming guidelines specific to slave mode
- DMA application information

45.7.1 Recommended interrupt service flow

The following figure shows a flowchart for the recommended I^2C interrupt service routine. Deviation from the flowchart may result in unpredictable I^2C bus behavior.

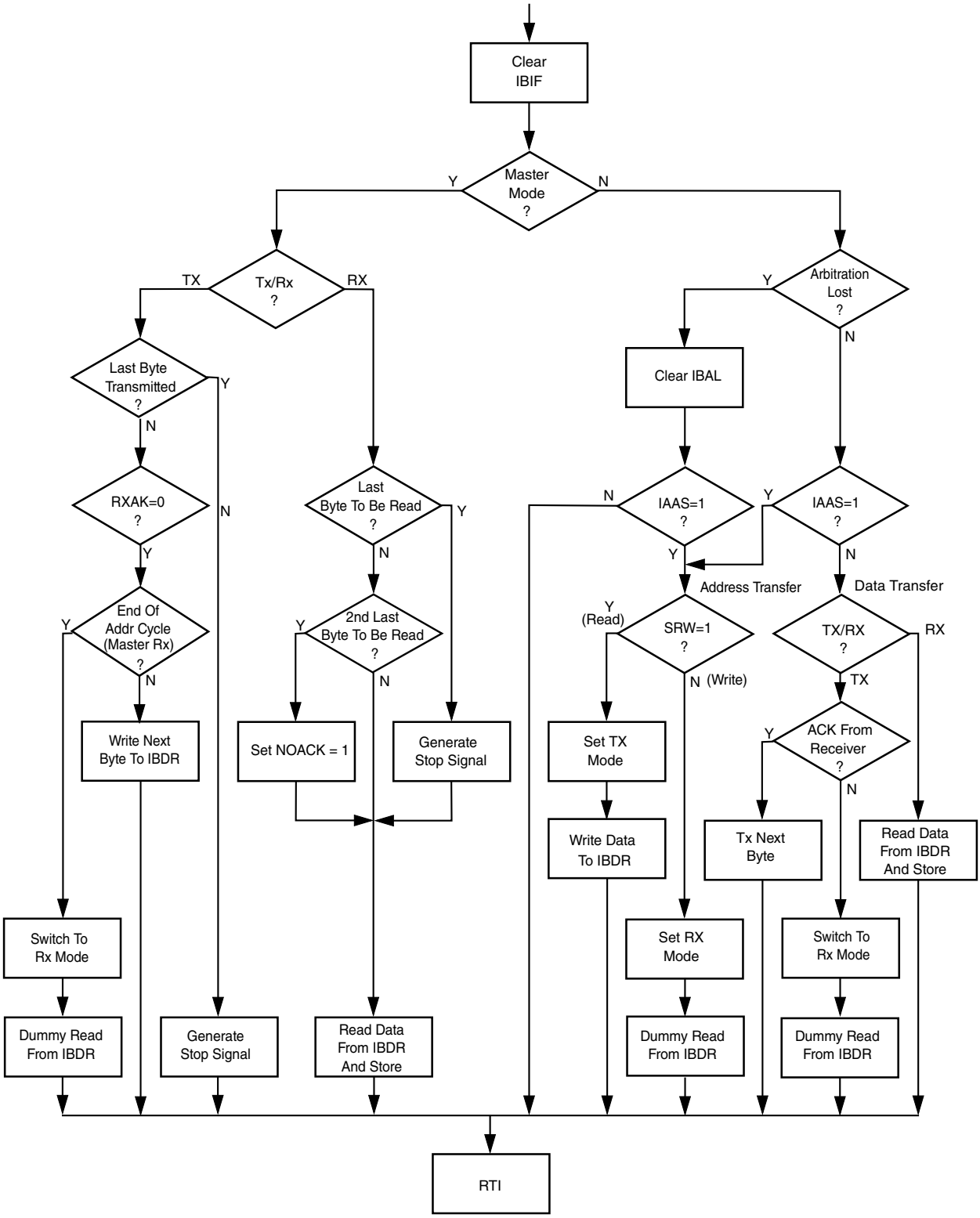


Figure 45-13. Recommended I²C interrupt service routine flowchart

45.7.2 General programming guidelines (for both master and slave mode)

This section provides programming guidelines recommended for the I2C module in both master and slave mode. It presents the following topics:

- Initializing the I2C module
- Software response after a transfer

45.7.2.1 Initializing the I2C module

The following table describes how to initialize the I2C module.

Table 45-11. I2C initialization procedure

Step	Action
1	Use I2C Bus Frequency Divider Register (I2C_IBFD) to select the required division ratio to obtain SCL frequency from the system clock.
2	Use I2C Bus Address Register (I2C_IBAD) to define the slave address.
3	Clear the MDIS field in I2C Bus Control Register (I2C_IBCR) to enable the I2C interface system.
4	Use I2C Bus Control Register (I2C_IBCR) to select Master/Slave mode, Transmit/Receive mode, and whether interrupts are enabled or disabled.
5	(Optional) Use I2C Bus Interrupt Config Register (I2C_IBIC) to further refine the interrupt behavior.

45.7.2.2 Software response after a transfer

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The I2C Bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. The IBIF (interrupt flag) can be cleared by writing one (in the interrupt service routine, if interrupts are used).

The TCF bit will be cleared to indicate data transfer in progress whenever data register is written to in transmit mode, or during reading out from data register in receive mode. The TCF bit should not be used as a data transfer complete flag as the flag timing is dependent on a number of factors including the I2C bus frequency. This bit may not conclusively provide an indication of a transfer complete situation. It is recommended that transfer complete situations are detected using the IBIF flag.

Software may service the I^2C I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Polling should monitor the IBIF bit rather than the TCF bit since their operation is different when arbitration is lost.

When a "Transfer Complete" interrupt occurs at the end of the address cycle, the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit sent with slave calling address, then the Tx/Rx bit at Master side should be toggled at this stage. If Master does not receive an ACK from Slave, then transmission must be re-initiated or terminated.

In slave mode, IAAS bit will get set in IBSR if Slave address (IBAD) matches the Master calling address. This is an indication that Master-Slave data communication can now start. During address cycles (IBSR[IAAS]=1), the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly. For slave mode data cycles (IBSR[IAAS]=0), the SRW bit is not valid. The Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

45.7.3 Programming guidelines specific to master mode

This section presents the following topics:

- Generating START
- Transmit/receive sequence
- Generating STOP
- Generating repeated START
- Loss of arbitration

45.7.3.1 Generating START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the I^2C Bus Busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB, which is set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I^{2}C is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of the sequence of events which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
while (IBSR[IBB]==1)                // wait in loop for IBB flag to clear
IBCR[MS/MSL] and IBCR[]Tx/Rx] = 1    // master and transmit mode, that is,
                                    // generate start condition
IBDR = calling_address              // send the calling address to the data register
while (bit 5, IBSR ==0)             // wait in loop for IBB flag to be set
```

45.7.3.2 Transmit/receive sequence

The following tables present the sequences for:

- Master transmit
- Master receive
- Slave transmit
- Slave receive

Table 45-12. Master transmit sequence

Step	Action
a	Use I2C Bus Frequency Divider Register (I2C_IBFD) to select the required division ratio to obtain SCL frequency from Platform clock/2.
b	Write 0 to the IBDIS field in I2C Bus Control Register (I2C_IBCR) to enable the I^{2}C interface system.
c	Use I2C Bus Control Register (I2C_IBCR) to select Master mode, Transmit mode, and interrupt enable.
d	Write 0 to the IBIF field in I2C Bus Status Register (I2C_IBSR).
e	Write data to I2C Bus Data I/O Register (I2C_IBDR).
f	Observe changes in the TCF field of I2C Bus Status Register (I2C_IBSR) : • When IBSR[TCF] becomes 0, the transfer is in progress. • When IBSR[TCF] becomes 1, the transfer is complete.
g	Wait until the IBIF field in I2C Bus Status Register (I2C_IBSR) becomes 1.
h	Read the fields in I2C Bus Status Register (I2C_IBSR) to determine what happened: • If TCF = 1, the transfer completed. • If RXAK = 1, a No Acknowledge condition occurred. • If IBB = 0, the bus transitioned from Busy to Idle state. • If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **NOTE:** You can ignore Address Detect (IAAS = 1) for master mode (it is valid only for slave mode).
i	Examine the RXAK field in I2C Bus Status Register (I2C_IBSR) for an acknowledgment from the slave.
j	Repeat steps d through i to transfer the next consecutive bytes of data.

Table 45-13. Master receive sequence

Step	Action
a	Follow steps a through i in Table 45-12 for address dispatch.
b	Write 0 to the IBIF field in I2C Bus Status Register (I2C_IBSR).
c	Write 0 to the TXRX field in I2C Bus Control Register (I2C_IBCR) to select Receive mode.
d	Perform a dummy read of I2C Bus Data I/O Register (I2C_IBDR) to initiate the receive operation.
e	Wait until the TCF field in I2C Bus Status Register (I2C_IBSR) becomes 1. (This proves that the transfer is complete.)
f	Wait until the IBIF field in I2C Bus Status Register (I2C_IBSR) becomes 1.
g	Read the fields in I2C Bus Status Register (I2C_IBSR) to determine what happened: • If TCF = 1, the reception completed. • If IBB = 0, the bus transitioned from Busy to Idle state. • If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **NOTE:** You can ignore the No Acknowledge condition (RXAK = 1) for receive mode.
h	Read I2C Bus Data I/O Register (I2C_IBDR) to determine the data received from the slave.

Table 45-14. Slave transmit sequence

Step	Action
a	Use I2C Bus Address Register (I2C_IBAD) to define the slave address.
b	Write 0 to the IBDIS field in I2C Bus Control Register (I2C_IBCR) to enable the I^2C interface system.
c	Examine fields in I2C Bus Status Register (I2C_IBSR) as follows: • If IAAS = 1, examine IBSR[SRW]. • If IAAS = 1 and SRW = 1, write 1 to IBCR[TXRX] to select Transmit mode.
d	Write data to I2C Bus Data I/O Register (I2C_IBDR).
e	Wait until the IBIF field in I2C Bus Status Register (I2C_IBSR) becomes 1.
f	Wait until the RXAK field in I2C Bus Status Register (I2C_IBSR) becomes 0.
g	Write 0 to the IBIF field in I2C Bus Status Register (I2C_IBSR).
h	Repeat steps d through g for the next consecutive data transfers.

Table 45-15. Slave receive sequence

Step	Action
a	Use I2C Bus Address Register (I2C_IBAD) to define the slave address.
b	Write 0 to the IBDIS field of I2C Bus Control Register (I2C_IBCR) to enable the I^2C interface system.
c	Examine fields in I2C Bus Status Register (I2C_IBSR) as follows: • If IAAS = 1, examine IBSR[SRW]. • If IAAS = 1 and SRW = 0, write 0 to IBCR[TXRX] to select Receive mode.
d	Write 0 to the IBIF field of I2C Bus Status Register (I2C_IBSR).
e	Perform a dummy read of I2C Bus Data I/O Register (I2C_IBDR) to initiate the receive operation.
f	Wait until the TCF field of I2C Bus Status Register (I2C_IBSR) becomes 1. (This proves that the transfer is complete.)
g	Wait until the IBIF field of I2C Bus Status Register (I2C_IBSR) becomes 1.

Table continues on the next page...

Table 45-15. Slave receive sequence (continued)

Step	Action
h	Read the fields in I2C Bus Status Register (I2C_IBSR) to determine what happened: • If TCF = 1, the reception completed. • If IBB = 0, the bus transitioned from Busy to Idle state. • If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **NOTE:** You can ignore the No Acknowledge condition (RXAK = 1) for receive mode.
i	Read I2C Bus Data I/O Register (I2C_IBDR) to determine the data received from the master.

45.7.3.3 Generating STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a STOP condition is generated by a master transmitter.

```
if (tx_count == 0) or      // check to see if all data bytes have been transmitted
  (bit 0, IBSR == 1) {     // or if no ACK generated
    clear bit 5, IBCR      // generate stop condition
  }
else {
 IBDR = data_to_transmit   // write byte of data to DATA register
   tx_count --             // decrement counter
 }                         // return from interrupt
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the NOACK bit in IBCR before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must first be generated. The following is an example showing how a STOP signal is generated by a master receiver.

```
rx_count --             // decrease the rx counter
if (rx_count ==1)       // 2nd last byte to be read ?
 bit 3, IBCR = 1        // disable ACK
 if (rx_count == 0)     // last byte to be read ?
 bit 5, IBCR = 0        // generate stop signal
else
data_received = IBDR    // read RX data and store
```

45.7.3.4 Generating repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
bit 2, IBCR = 1             // generate another start (restart)
IBDR == calling_address     // transmit the calling address
```

45.7.3.5 Loss of arbitration

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices that lost arbitration are immediately switched to slave mode by the hardware. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/SL=0. If one master attempts to start transmission, while the bus is being engaged by another master, the hardware will inhibit the transmission, switch the MS/SL bit from 1 to 0 without generating a STOP condition, generate an interrupt to CPU, set the IBAL to indicate that the attempt to engage the bus is failed, and not set the TCF due to the loss of data during arbitration. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

45.7.4 Programming guidelines specific to slave mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred. Interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR for slave transmits or dummy reading from IBDR in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an "end of data" signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

45.7.5 DMA application information

The DMA interface on the I^2C is not completely autonomous and requires intervention from the CPU to start and to terminate the frame transfer. DMA mode is only valid for Master transmit and Master receive modes. Software must ensure that the DMAEN field in I2C Bus Control Register (I2C_IBCR) is not set when the I^2C module is configured in slave mode.

The DMA controller must only transfer one byte of data per Tx/Rx request. This is because there is no FIFO on the I^2C block.

The CPU should also keep the I^2C interrupt enabled during a DMA transfer to detect the arbitration lost condition and take action to recover from this situation. The DMAEN field in I2C Bus Control Register (I2C_IBCR) works as a disable for the transfer complete interrupt. This means that during normal transfers (no errors) there will always be either an interrupt or a request to the DMA controller, depending on the setting of the DMAEN field. All error conditions will trigger an interrupt and require CPU intervention. The address match condition will not occur in DMA mode as the I^2C should never be configured for slave operation.

The following sections detail how to set up a DMA transfer and what intervention is required from the CPU. It is assumed that the system DMA controller is capable of generating an interrupt after a certain number of DMA transfers have taken place.

The sections present the following topics:

- DMA mode, master transmit
- DMA mode, master reception
- Exiting DMA mode, system requirement considerations

45.7.5.1 DMA mode, master transmit

The following flow diagram details exactly the operation for using a DMA controller to transmit "n" data bytes to a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the last data byte) can be transferred by the DMA controller. The last data byte must be transferred by the CPU.

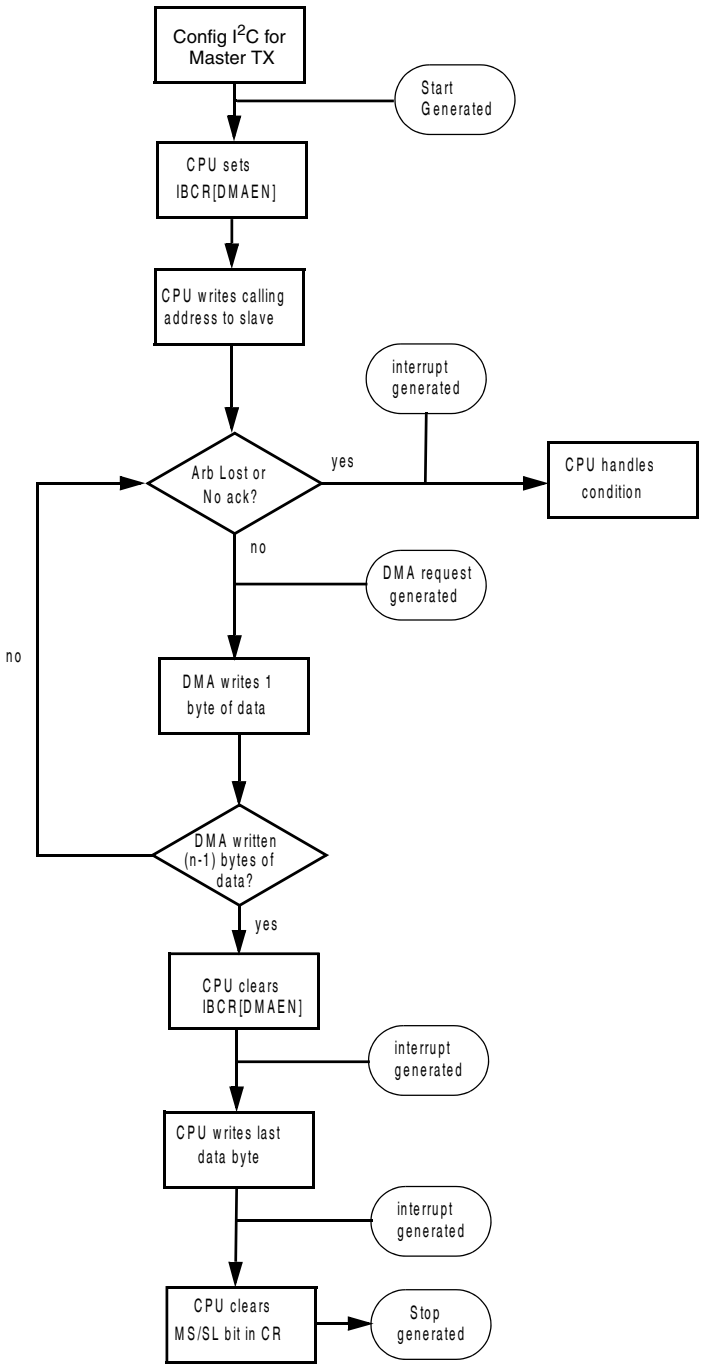


Figure 45-14. Flow-Chart of DMA mode master transmit

45.7.5.2 DMA mode, master reception

The following flow diagram details the exact operation for using a DMA controller to receive "n" data bytes from a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the two last data bytes) can be read by the DMA controller. The last two data bytes must be transferred by the CPU.

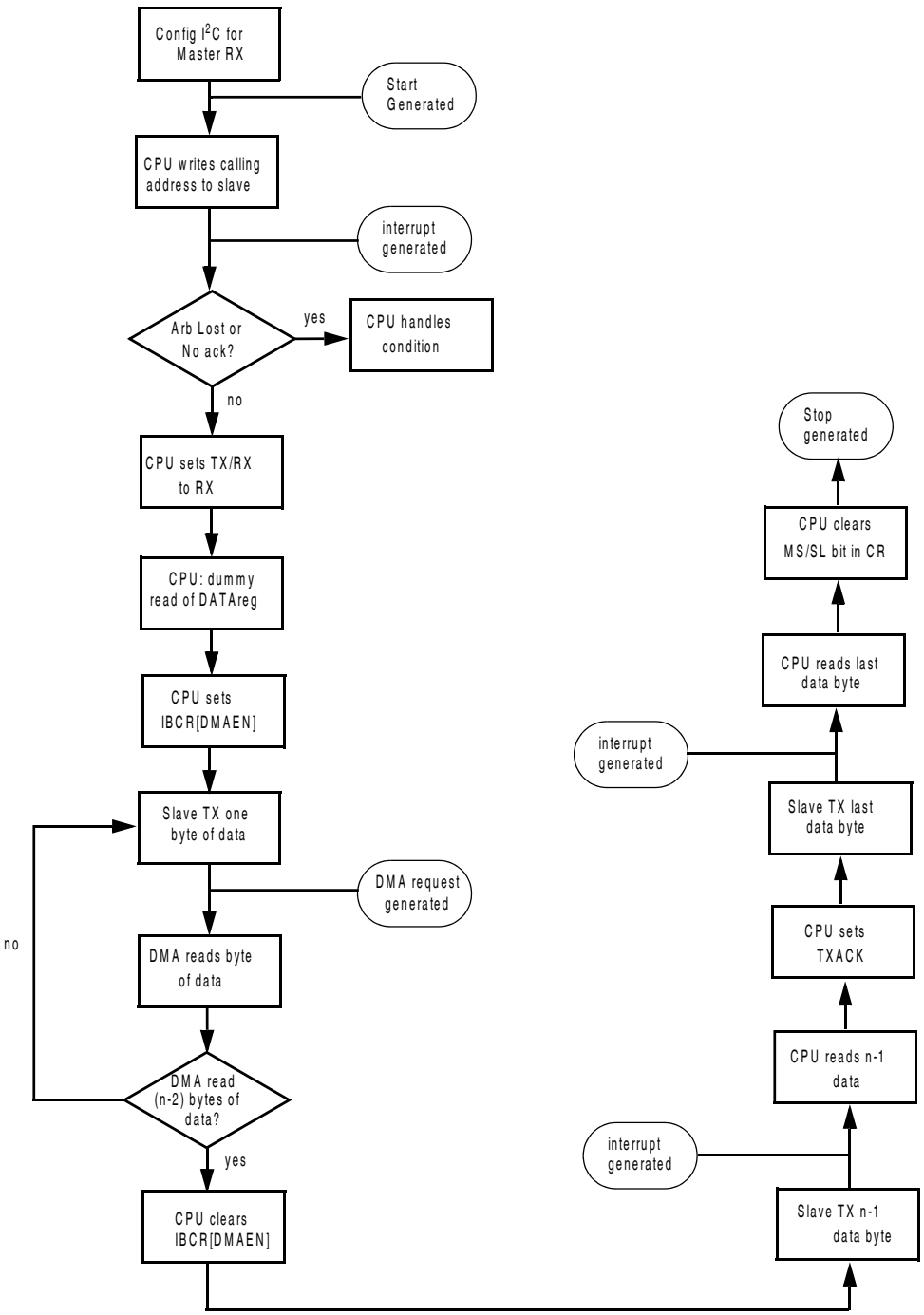


Figure 45-15. Flow-Chart of DMA mode master receive

45.7.5.3 Exiting DMA mode, system requirement considerations

As described above, the final transfers of both TX and RX transfers need to be handled via interrupt by the CPU. To change from DMA to interrupt driven transfers in the I^2C module, you have to disable the DMAEN bit in the IBCR register. The trigger to exit the DMA mode is that the programmed DMA Transfer Control Descriptor (TCD) has completed all its transfers to/from the I^2C module.

After the last DMA write (TX mode) to the I^2C the module will immediately start the next I^2C-bus transfer. The same is true for receive mode. After the DMA read from the IBDR register the module initiates the next I^2C-bus transfer. This results in two possible scenarios in the DMA mode exiting scheme.

1. Fast reaction

   The DMAEN bit is cleared before the next I^2C-bus transfer completes. In this case the module will raise an interrupt request to the CPU which can be serviced normally.

2. Slow reaction

   The DMAEN bit is cleared after the next I^2C-bus transfer has already completed. In this case, the module will not raise an interrupt request to the CPU. Instead the TCF bit can be read to determine that the transfer completed and the module is ready for further transfer.

What is fast/slow reaction?

The reaction time T_R for the system to disable DMAEN after the last DMA controller access to the I^2C is the time required for one byte transfer over the I^2C. For 'fast reaction' the disabling has to occur before the 9^{th} bit of the data transfer which is the ACK bit. So the time available is eight times the SCL period.

$$T_R = 8 \times T_{SCL}$$

In fast mode, with 400kbit/s, T_{SCL} is 2.5µs, so T_R is 20µs.

Depending on the system and DMA controller there are different possibilities for the de-assertion of DMAEN. Three options are:

1. CPU intervention via Interrupt

   The DMA controller is programmed to signal an interrupt to the CPU which is then responsible for the de-assertion of DMAEN. This scheme should be supported by most systems but it can result in a slow reaction time if other higher priority

interrupts interfere. Therefore the interrupt handling routine can become complicated as it has to check which of the two cases happened (check TCF bit) and act accordingly. In case of slow reaction you can force an interrupt for the I^2C in the interrupt controller to have the further transfer handled by the normal I^2C interrupt routine.

Note

The use of nested interrupts can still cause potential issues in this scenario, if someone tries to stall the DMA interrupt between the de-assertion and DMAEN bit and checks the TCF bit.

2. DMA channel linking

   If the Transfer control descriptor in the DMA controller that performs the data transfer is linked to another channel that does a write to IBCR to disable the DMAEN field, this might probably be the fastest system solution, but it uses two DMA channels.

   Note

   Here you have to make sure on system level that no higher priority DMA requests occur between the two linked TCDs as those could again create a scenario of slow reaction.

3. DMA scatter/gather process

   If the Transfer control descriptor in the DMA controller that performs the data transfer has the scatter/gather feature activated, this feature will initiate a reload of another TCD from system RAM after the completion of the first TCD. The new TCD will have its start bit already set and immediately start the required write to the IBCR to disable the DMAEN field. This TCD also has scatter/gather activated and is programmed to reload the initial TCD upon completion, bringing the system back into a "ready-for-I^2C-transfer" state. The advantage over the two other solutions is that this requires neither CPU intervention nor a second DMA channel. This comes at the cost of 64 bytes RAM (two TCDs), some system bus transfer overhead and a little increase in application code complexity.

Note

Here you have to make sure at system level that no higher priority DMA requests occur during the scatter/gather process, as those could again create a scenario of slow reaction.

Example latencies for a 32 MHz system with a full speed 32-bit AHB bus and an I^2C connected via half speed IPI bus:

- Accessing the I^2C from the DMA controller via IPI bus typically requires four cycles (consecutive accesses to the I^2C could be faster):

  $4 \times T_{IPI}$ = 4/16 MHz = 250 ns

- Reloading a new TCD (8 x 32-bit) via AHB to the DMA controller (scatter/gather process):

  $8 \times T_{AHD}$ = 8/32 MHz = 250 ns

Example latencies for a 150 MHz system with a full speed 32-bit AHB bus and an I^2C connected via half speed IPI bus:

- Accessing the I^2C from the DMA controller via IPI bus typically requires four cycles (consecutive accesses to the I^2C could be faster):

  $4 \times T_{IPI}$ = 4/150 MHz = 26.6 ns

- Reloading a new TCD (4 x 64-bit) via AHB to the DMA controller (scatter/gather process):

  $4 \times T_{AHD}$ = 4/150 MHz = 26.6 ns

With the DMA scatter/gather process the required IBCR access can be done in 0.5 µs, leaving a large margin of 19.5 µs for additional system delays. In this way, the slow reaction case can be prevented. The system user needs to decide which usage model best suits his overall requirement.

Chapter 46
LINFlexD

46.1 Chip-specific LINFlexD information

46.1.1 LINFlexD configurations

The device contains 16 LINFlexD instances. The following table shows the instantiation information for each module.

Table 46-1. LINFlexD configurations

Description	LINFlexD[0]	LINFlexD[1]	LINFlexD[2....15]
Number of filters implemented (Identifier Filter Control Register (LINFlexDx_IFCRn))	16	0	0
Number of Tx DMA channels	1	1	0
Number of Rx DMA channels	1	1	0
LIN operation mode	Master and Slave	Master only	Master only
Autosyncronization	Yes	No	No
DMA Support	Yes	Yes	No

UART Support - LINFlexD modules support UART operation up to 1 Mbps.

The LINFlex modules implemented on MPC5746C do not support communication with Special Word Length in UART mode set (UARTCR[WLS]=1) in buffer mode or in FIFO mode with UARTCR[WL1, WL0]=0,1.

NOTE

When LINFlex is in idle state the TX pins are set to high-impedance. If LIN transceiver does not have internal pull-up an external pull-up may be required.

LINFlexD expects the RX pin to be high whenever it is enabled via programming of MC_ME_PCTL. In case the SIUL programming is done after PCTL is enabled, WUF flag of LINSR register would be set.

46.1.2 LinflexD's stop-stop ack protocol

1. An acknowledgement to stop request is sent only after the current transfer/reception or DMA transfer request of the programmed data bytes is completed and the state machine goes to idle.
2. If the module reset is asserted the stop request is acknowledged immediately.

46.2 Introduction

The LINFlexD controller is designed to manage a high number of LIN messages efficiently with a minimum of CPU load. To reduce the CPU loading in Master mode, LINFlexD autonomously handles the LIN messages once software has triggered the header transmission, until the next header transmission request in transmitter mode or until checksum reception in the receiver mode.

The LINFlexD supports LIN protocol version 1.3, 2.0, and 2.1. It also consists of an 8-byte buffer for transmission/reception data.

The LINFlexD also provides support for some of the basic UART transfers of 8-bit, 9-bit, 16-bit, and 17-bit frames and also 12-bit data frame + parity reception in UART mode for MSC support.

The LINFlexD supports multi-channels and parametric DMA Tx/Rx interface in LIN/UART operating mode.

46.2.1 Glossary and acronyms

Table 46-2. Glossary and acronyms

Term	Description
LIN	Local interconnect network
UART	Universal asynchronous transmitter receiver
ID	LIN identifier field
DMA	Direct memory access
TCD	Transfer control descriptor
IPS	Peripheral Bus Interface

Table continues on the next page...

Table 46-2. Glossary and acronyms (continued)

Term	Description
IRQ	Interrupt request
FSM	Finite state machine
WS	Wait state
SW	Software
CPU	Central processing unit
SoC	System on a chip

46.2.2 References

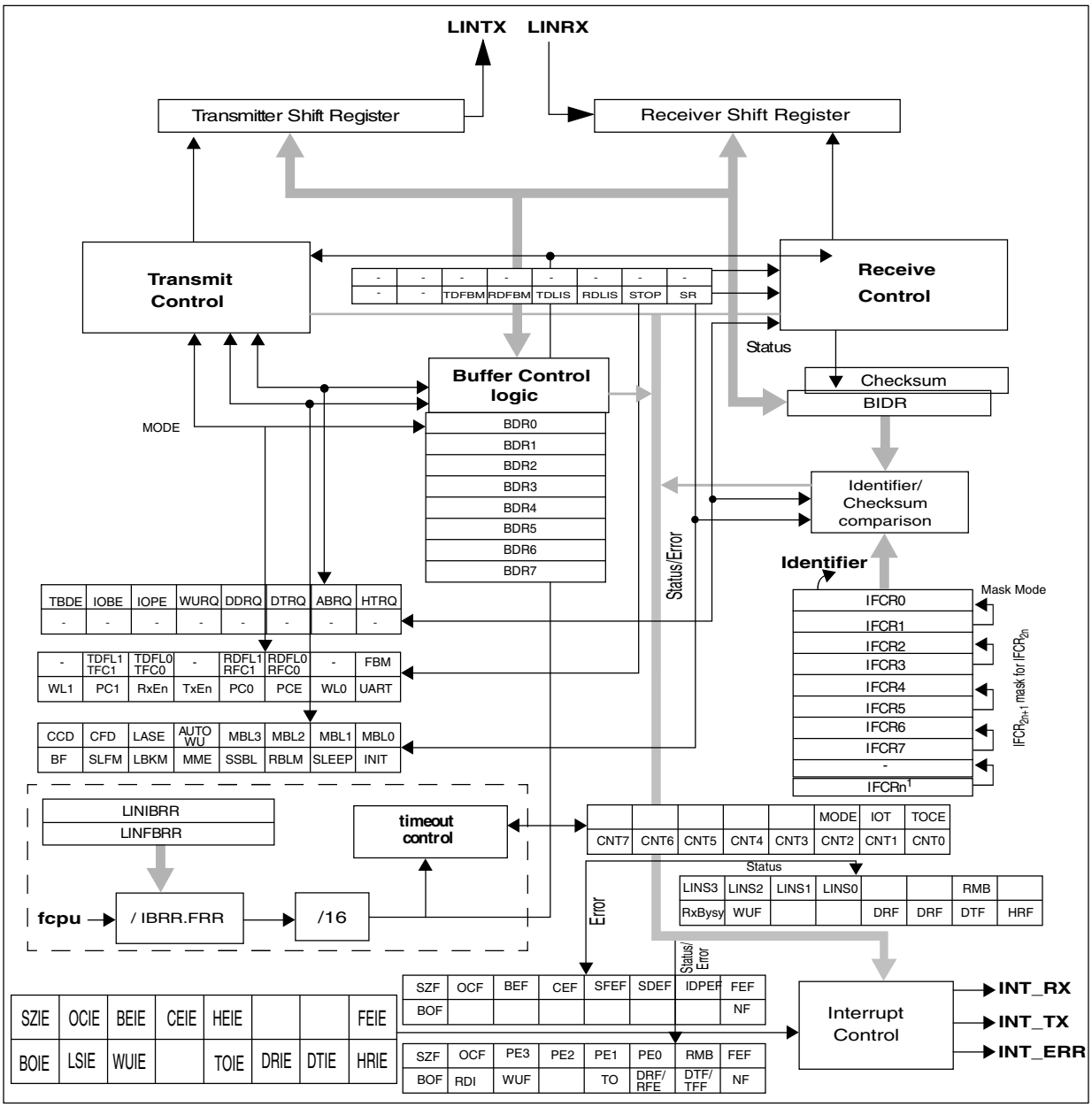


Figure 46-1. Block diagram

Notes:

1: The value of n depends on the no_of_filters. Refer to the chip configuration details to see the number of filters used in this device.

2: For LIN mode N = 16 and N is programmable by user in UART mode.

46.3 Main features

The LINFlexD controller can operate in several modes, each of which has a distinct set of features that are described in the following sections. In addition, the LINFlexD controller has several features common to all modes:

- Fractional baud rate generator
- 3 operating modes for power saving and configuration registers lock
  - Initialization
  - Normal
  - Sleep
- Test Mode:Loop Back
- Maskable interrupts
- A maximum of 16 possible identifiers can be programmed into the identifier list

46.3.1 LIN mode features

- Supports LIN protocol version 1.3, 2.0, and 2.1
- Bit rates up to 20 Kbit/s (LIN protocol)
- Master/Slave mode
- Classic and Enhanced Checksum calculation and check
- Single 8-byte buffer or FIFO for Transmission/Reception
- Timeout management
- Identifier filters
- DMA interface
- Supports a maximum of 16 possible identifiers
- Master mode with autonomous message handling
- Wakeup event on dominant bit detection
- True LIN field state machine

- Advanced LIN error detection
- Header, response, and frame timeout
- Slave mode
  - Autonomous header handling
  - Autonomous transmit/receive data handling
- Identifier filters for autonomous message handling in Slave mode
- Separate clock for baud rate calculation
  - The relationship “(2/3)* LIN_CLK > PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained.

46.3.2 UART mode features

- Full-duplex communication
- Baud rate is a function of baud clock, LINIBRR and LINFBRR registers - see Baud rate generation
- Separate clock for baud rate calculation
  - The relationship “(2/3)* LIN_CLK > PBRIDGEx_CLK > 1/3*LIN_CLK” should be maintained.
- 15/16/7/8 bits data, parity
- 1/2/3 stop bits
- 12-bit + parity reception
- 4-byte buffer for reception, 4-byte buffer for transmission
- 12-bit counter for timeout management
- The maximum baud rate achievable is 25 Mbit/s.
- For bit rate ≤ 6.25 Mbit/s
  - Sixteen times oversampling
  - 3:1 majority voting
- For 6.25 Mbit/s < bit rate ≤ 12.5 Mbit/s

- Reduced over sampling programmable by the user
- 3:1 majority voting for reduced over sampling of 8

- For 12.5 Mbit/s < bit rate ≤ 25 Mbit/s
  - Reduced over sampling programmable by the user
  - 1:1 voting for all reduced over sampling of 4, 5 and 6

46.4 Functional description

46.4.1 LIN protocol

The LIN (Local Interconnect Network) is a serial communication protocol. A LIN cluster consists of one master task and several slave tasks. A master node contains the master task as well as a slave task. All other nodes contain a slave task only. The master node decides when and which frame is transferred on the bus. The slave task provides the data to be transported by the frame.

46.4.1.1 Frames

A frame consists of a header provided by the master task and a response provided by the slave task. The header consists of a break, a sync pattern, and an identifier. The break is followed by the sync pattern and the sync pattern is followed by the identifier. The slave task associated with the identifier provides the response. The response consists of a data field and checksum. The slave task designated to receive the data associated with the identifier receives the response and verifies the checksum.

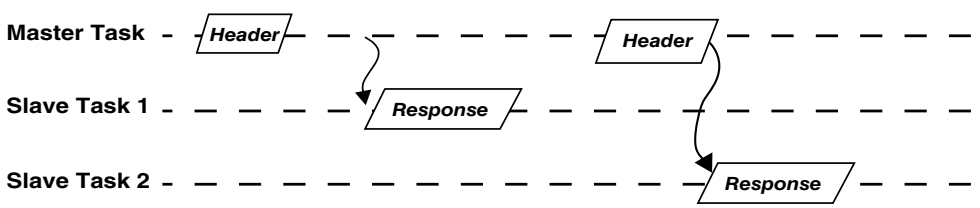


Figure 46-2. Frames

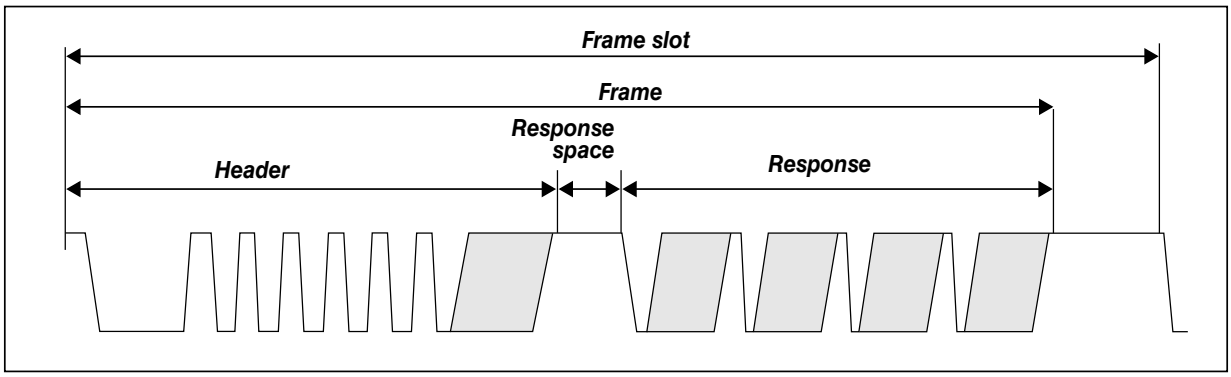


Figure 46-3. Structure of LIN frame

46.4.1.2 Data field

Each byte is transmitted as shown in Figure 46-4. The LSB of the data is sent first and the MSB is sent last. The start bit is encoded as a bit with value zero (dominant) and stop bit is encoded as bit value one (recessive).

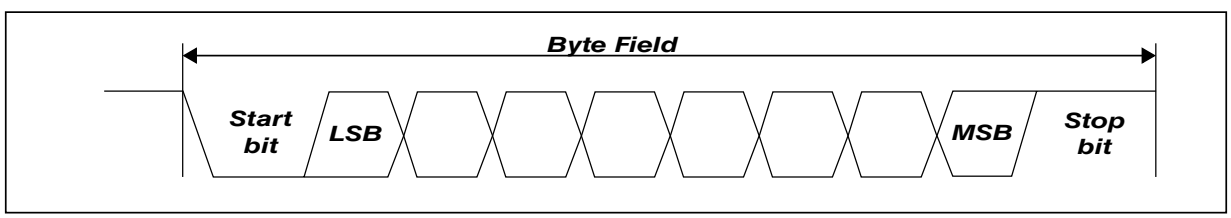


Figure 46-4. Structure of byte field

46.4.1.3 Break

The break symbol is used to signal the beginning of a new frame. It is the only field that does not comply with the above figure. The break is always generated by the master and shall be at least 13 bits of dominant value including the start bit, followed by a break delimiter as shown in Figure 46-5. The break delimiter must be of two-bit duration (to be compliant with LIN protocol 2.1).

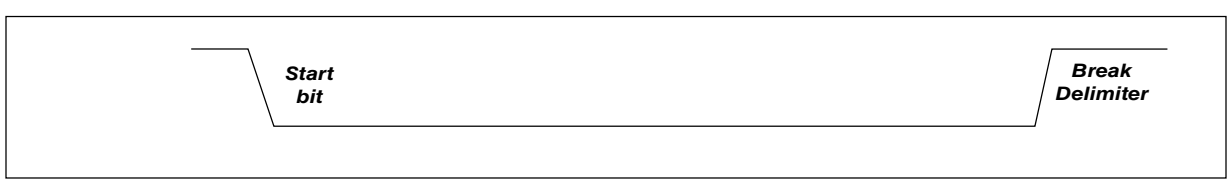


Figure 46-5. Break field

46.4.1.4 Sync byte

Sync is a byte field with the data value of 0x55.

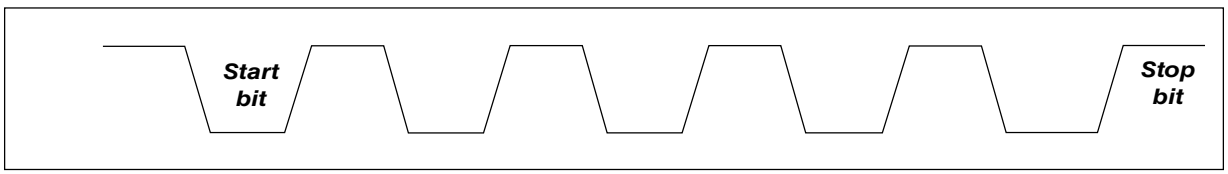


Figure 46-6. Sync byte field

46.4.1.5 Identifier

The Identifier field consists of two sub-fields, the identifier and the identifier parity. Bits 0 to 5 are the identifier and bits 6 and 7 indicate the parity.

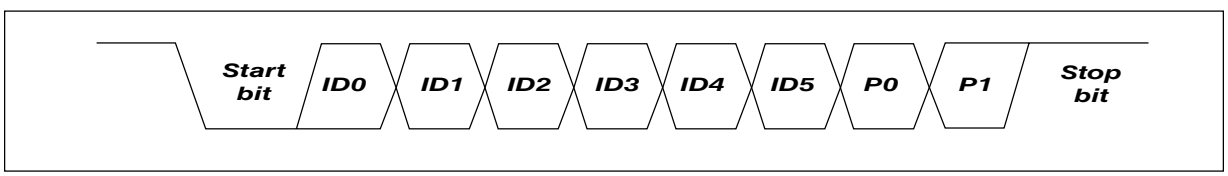


Figure 46-7. Identifier field

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

46.4.1.6 Checksum

The last field of a frame is the checksum. The checksum contains the inverted 8-bit sum with carryover of all data bytes or all data bytes and the identifier. Checksum calculation over the data bytes only is called classic checksum and is used for communication with LIN 1.3 slaves. Checksum calculation over the data bytes and the protected identifier byte is called enhanced checksum and is used for communication with LIN 2.0 slaves.

46.4.2 LINFlexD features

46.4.2.1 Operating modes

The LINFlexD has three operating modes: **Initialization, Normal** and **Sleep** modes. After hardware reset, the LINFlexD enters sleep mode to reduce power consumption.

Initialization mode (INIT)

To enter this mode, software sets the INIT bit in the LINCR1. To exit the initialization mode, software should reset the INIT bit.

When in initialization mode, all message transfers to and from the LIN bus are stopped and the status of LIN bus output LINTX is recessive (high). If software invokes the initialization mode when a bus transfer is in progress, the transfer is aborted. The software should therefore check the LIN state before setting this bit.

To initialize the LINFlexD controller software must:

1. Set up the baud rate registers
2. Reset UART bit
3. Select the mode (master or slave)
4. Configure checksum control bits
5. Initialize the identifier list (Slave mode)

Normal mode (NM)

Once software has completed initialization of the LINFlexD controller, it can enter the Normal mode by clearing the INIT bit.

Sleep mode (SM)

Sleep mode in LINFlexD reduces power consumption. This mode is entered by setting the SLEEP bit in LINCR1. In this mode the LINFlexD clock is stopped. LINFlexD can be awakened from Sleep mode by clearing the SLEEP bit.

If software detects a wakeup pulse of 150 µs on the LIN Bus, it can request LINFlexD to wake up from Sleep mode. Refer to Wakeup management.

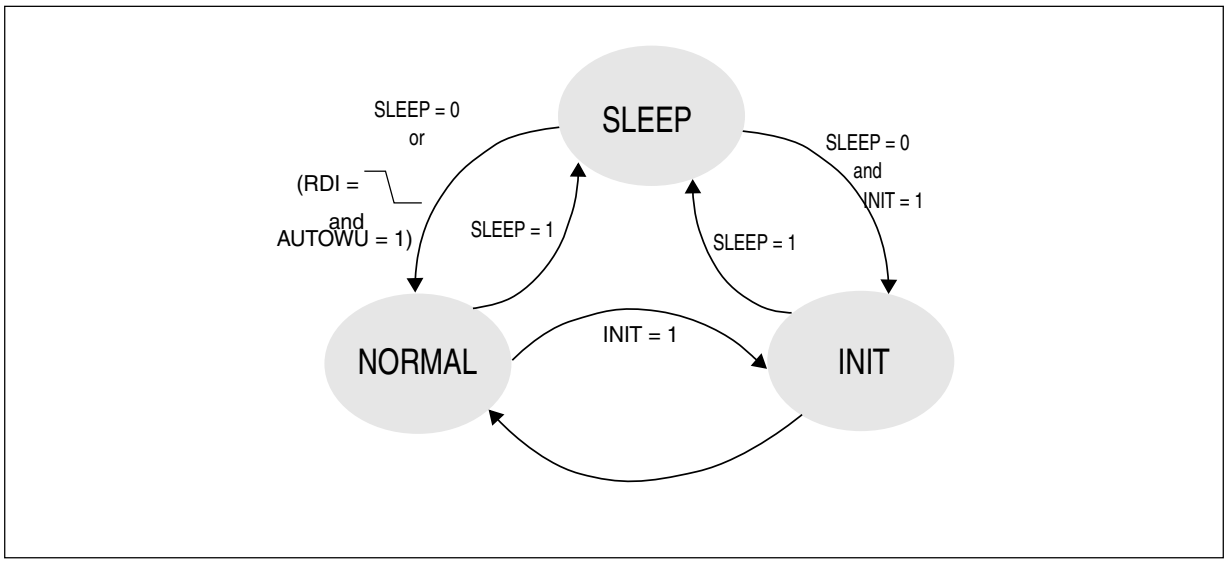


Figure 46-8. Operating modes

46.4.2.2 Test mode

Loop Back mode

This mode is entered by setting the LBKM bit in LINCR1. In this mode, the LINFlexD receives the Identifier and Data transmitted by itself and writes the same in the BIDR and Data buffers respectively. This mode is provided for selftest functions. Bit error is checked in this mode.

The LINFlexD ignores the LINRX signal. There is an internal feedback from its TX output to its RX input. The TX pin can be disconnected from LINTX pin by SIUL2 setting.

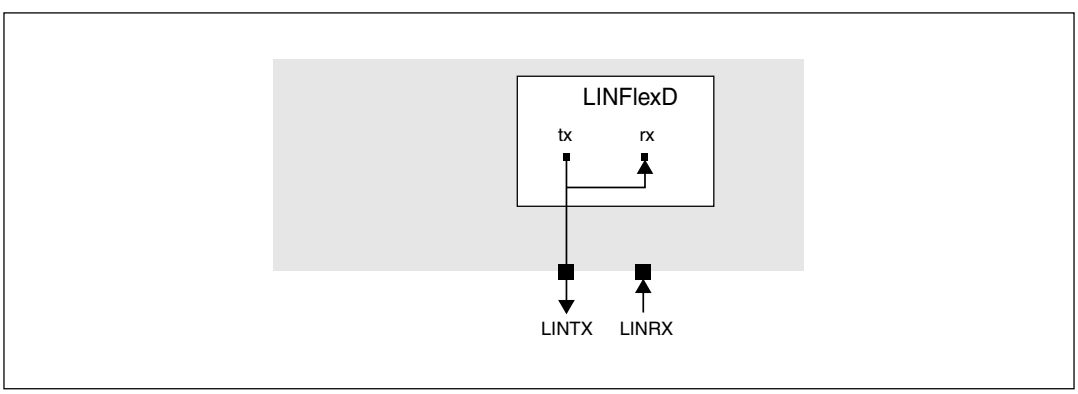


Figure 46-9. LINFlexD in Loop Back Mode

46.4.2.3 Master mode

Master mode is selected by means of the MME bit in LINCR1 register.

Header transmission

According to the LIN protocol any communication on the LIN bus is triggered by the master sending a header. The header is transmitted by the master task of a node while the response is transmitted by the slave task of a node.

To transmit the header, first program the Identifier, data field length, message direction and checksum enable in the BIDR register; then set the HTRQ bit in LINCR2. Once header transmission starts, the user should not modify BIDR register bits until the current frame is complete. The transmitted ID is also received by the master node and copied to BIDR.

Data transmission

When the master node is the publisher of the data corresponding to the Identifier sent by the master, then the slave task of the node should send the data in the response part of the frame. Hence software must provide the data to the LINFlexD before the header transmission is requested. The data to be transmitted is stored in the message buffer BDRL and BDRM. The number of bytes to be transmitted depends on the Data Field length in BIDR. The software uses the BIDR[CCS] bit to configure the checksum type (classic or enhanced) for each message.

If the response has been sent successfully, LINSR[DTF] is set. In case of error, the DTF flag is not set and the corresponding error flag is set in the LINESR (refer to error handling).

The direction of the message buffer is decided by the DIR bit in the BIDR. The transmitted data is also received by the same node and copied to the buffer.

Data reception

To receive data from a slave node, the master sends a header with the corresponding Identifier. The data received from the slave is stored in the message buffer and the status of the message is stored in the LINSR.

If the response has been received successfully, the LINSR (DRF) bit is set. In case of error, the DRF flag is not set and the corresponding error flag is set in the LINESR (refer to Error handling).

Data Discard

If the user wants to discard the data after header transmission, then the DDRQ bit in LINCR2 should be set.

46.4.2.4 Slave mode

This mode is selected when the MME bit of the LINCR1 register is cleared.

Data transmission

On header reception, the HRF bit is set and an RX interrupt is generated. The software must then:

1. Read the received ID in the BIDR register
2. Fill the BDR[0:x] register
3. Program the CCS and DIR bits in the BIDR register
4. Specify the data field length DFL bits in the BIDR register
5. Trigger the data transmission by setting the DTRQ bit

Note that the HRF bit should be reset only after the DTRQ bit is set. For the DTRQ to be effective, the HRF bit must be set. This is to ensure that DTRQ is not set randomly, but only after a header reception. It must be noted that you cannot set the DIR and DTRQ bits once RXbusy is asserted in LINSR.

Alternately, one or more Identifier filters are configured for transmission by setting the DIR bit, and activated by setting the enable bits in IFER. When at least one Identifier filter is active and configured for transmission and the received ID matches the filter, a TX interrupt is generated. The software can use the index in the IFMI register to point directly to the corresponding data array in the RAM and copy this data to the BDR[0:7].

The use of a filter saves software the processing time required to read the ID value in the BIDR, match it, and configure the data field length and checksum type.

If the number of filters provided by LINFlexD are not sufficient for the application, mask mode can be used for the filters.

The transmitted data is also received by the same node and copied to the buffer.

Data reception

When LINFlexD is the subscriber of the data of the received identifier, then on header reception the HRF flag is set and an RX interrupt is generated. The software must read the received ID from the BIDR register and specify the data field length before the reception of the stop bit of the first data byte. When the checksum is received, an RX

interrupt is generated for software to read the received data from BDR[0:7] and the RMB bit is set. Software must then release the data buffer by resetting the RMB bit in the LINSR.

When at least one identifier filter is active and configured for reception, an RX interrupt is generated only after the checksum reception. No interrupt request is generated on reception of the ID.

If the Identifier is filtered by software then you can discard the data by setting the DDRQ bit in the LINCR2, while HRF is set.

Note that for software filtering, software must decide the type of checksum (configure the CCS bit of the BIDR register) before reception of data. Otherwise the previous value of the CCS bit is considered while calculating checksum.

46.4.2.5 Errors

46.4.2.5.1 Header error

A header error is an error during the header reception. The error types are:

1. Sync Del error (SDEF)
2. Sync field error (SFEF)
3. Identifier Parity error (IDPEF)

Sync Del error (SDEF)

The delimiter should be 1 for at least one bit time; otherwise it is considered short and consequently the receiver discards synchronization on the current header. Hence the frame is discarded. An interrupt is generated if *HEIE* bit of LINEIER is set.

Sync field error (SFEF)

The SFEF error condition is monitored differently depending on whether autosynchronization (LASE) is ON or OFF.

Case 1: Autosynchronization ON (LASE bit of LINCR1 = 1):

When Autosynchronization is enabled, the SFEF flag indicates:

- The deviation error on the Sync Field is outside the LIN specification, which allows up to 14% of period deviation between the slave and master oscillators or
- An overflow has occurred during the Sync Field Measurement, which leads to an overflow of the divider registers.

Deviation error on the Sync Field

The deviation error is checked by comparing the current baud rate (relative to the slave oscillator) with the received LIN Sync Field (relative to the master oscillator).

This check is based on a measurement between the first falling edge and the last falling edge of the Sync Field. Let's refer to this period deviation as D.

If SFEF field is asserted it means that:

D > 14.0625%

If there is no error, it means that:

D < 14.84375%

If 14.0625% < D < 14.84375%, then the Sync Field could be either consistent or inconsistent depending on dephasing between the signal on the RDI line and the LIN_CLK.

Overflow during Sync Field Measurement

This check is based on the measurement of each bit time between both edges of the Sync Field. This checks that each of these bit times is large enough (more than 12 samples) compared to the bit time of the current baud rate.

Case 2: Autosynchronization OFF (LASE bit of LINCR1 = '0')

In this case the Sync character is received as a normal character. If the received character is 0x55 then the Sync Field is OK.

On any occurrence of an inconsistent Sync Field, the receiver immediately exits from Sync_Field state and the frame is consequently discarded. The SFEF bit of LINESR is set.

Identifier Parity error(IDPEF)

There are two parity bits in the identifier field. These bits are checked against the hardware-calculated parity over the other six bits of identifier, according to the formula:

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

This parity is checked after the ID is transferred to the BIDR register (after stop of ID has been detected properly) and upon parity mismatch, the receiver state machine exits from Identifier state immediately if the IOPE bit of LINCR2 is set.

46.4.2.5.2 Bit error

This error is flagged in transmission mode when the value read back from the bus is different from the value transmitted. Bit error checking on each bit is guaranteed if transceiver delay is less than one bit time minus 6 LIN_CPU cycles. Bit error is not checked during break field transmission.

1-bit time at 20 Kbit/s = 50 µs

6 LIN_CLK cycles at 80 MHz = 75 ns

Thus, (1-bit time) - (6 LIN_CLK cycles) = 49.925 µs

Transmission of the frame is stopped after the corrupted bit if the IOBE bit in LINCR2 is set. If IOBE is reset, the transmitter continues to transmit in spite of the bit error. An interrupt is generated if the BEIER bit is set in LINIER.

Note

If the break delimiter is not detected by the master within one bit time after delimiter transmission due to transceiver delay or error on the bus, then a train of bit error interrupts are generated. In this case, the Identifier may not be replaced in the BIDR.

Similarly, if the start of a falling edge of data is not detected by the transmitter node within one bit time after start bit transmission, then a train of bit error interrupts are generated. In this case also, data and checksum replacements in the BDR and CFR respectively are not guaranteed.

46.4.2.5.3 Framing error

This error is flagged when a dominant state is sampled on stop bit of the current received character (sync field, identifier field, data field, checksum field). LINFlexD discards the current frame and returns to Idle state. An interrupt is generated if FEIE bit is set in LINEIER.

The byte that caused framing error is also shifted to the buffer but DRF is never set in this case.

46.4.2.5.4 Checksum error

This error is flagged when the checksum computed by hardware does not match the received checksum.

LINFlexD discards the received frame and returns to Idle state. An interrupt is generated if the CEIE bit is set in LINEIER.

46.4.2.5.5 Overrun error

Once the message buffer is full (RMB is set), the next valid message reception will lead to an overrun and the message will be lost. The hardware signals the overrun condition by setting the BOF bit. Which message is lost depends on the buffer lock function control bit RBLM.

If RBLM is cleared, the old message in the buffer is overwritten by the most recent message. If RBLM bit is set, the most recent message is discarded and the oldest message is available to the software. In the case of slave, if buffer is not released (RMB is not reset) before reception of next Identifier and if RBLM is set, then the ID along with the data is discarded.

46.4.2.5.6 Timeout error

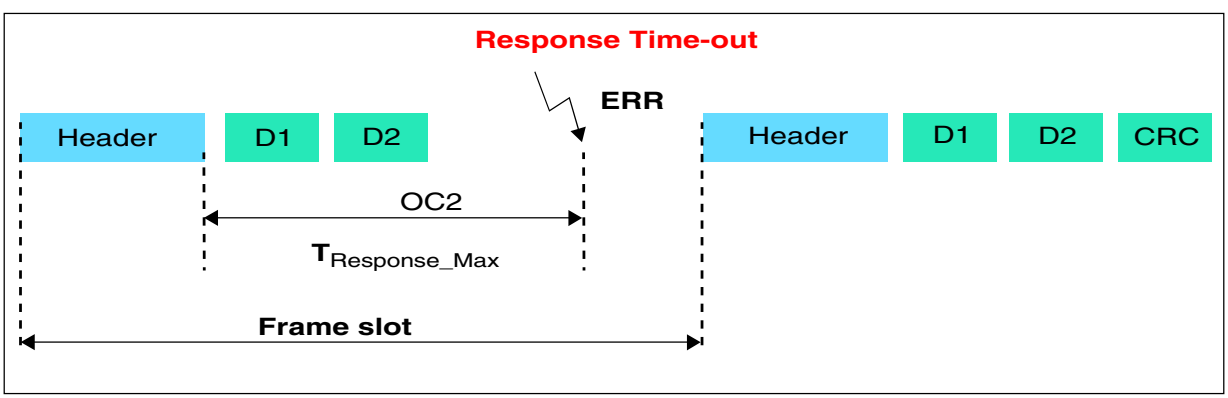


Figure 46-10. Incomplete response (for example, missing checksum)

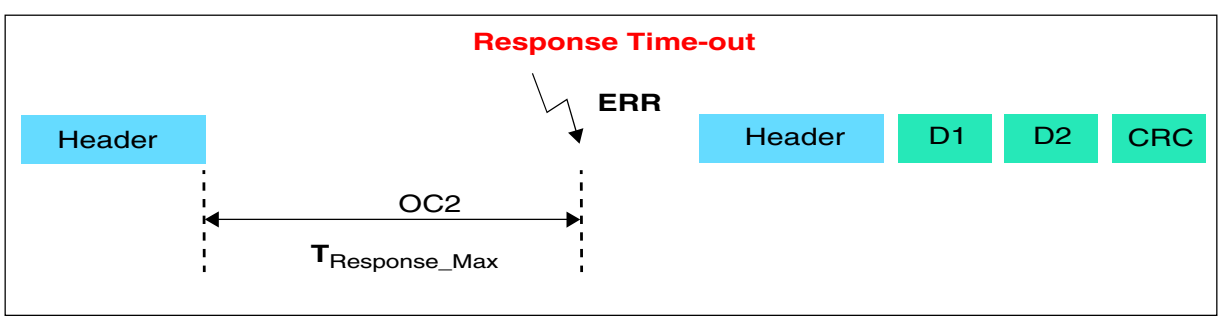


Figure 46-11. No response

Response timeout mechanism

- Master mode
  - OC2 is loaded with Nominal_time_out + LINFlexD_LINTCSR[CNT] at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time).

    This loading takes place at the end of id field (and not at the beginning of data field). Moreover, the correct value, which depends on DFL, is loaded immediately. No further OC2 update is required.
  - In case of no response at all within this time (in other words, start of data is not received), timeout takes place when the counter reaches the OC2 value (no change compared to earlier implementation).
  - In case of an incomplete response, timeout also occurs when the counter reaches the OC2 value (no change compared to earlier implementation).
- Slave mode
  - **Case 1**: The received identifier is managed by a filter. The implementation can be similar to the master mode, because the DFL value is loaded by hardware. Thus, OC2 can be loaded with Nominal_time_out + LINFlexD_LINTCSR[CNT] at the end of the stop bit of the Identifier field (where nominal_time_out = 1.4 × ((DFL + 2) × 10 bit time))
  - **Case 2**: The received identifier is not managed by a filter. As the DFL value needs to be updated by software after the identifier field has been received, the implementation is the following:
    - At the end of the ID, OC2 is loaded with 36 (maximum possible response space) + LINFlexD_LINTCSR[CNT].
    - At the end of the first_data_byte it is reloaded again according to DFL
    - Before reloading, LINFlexD checks the count_val. If count value is higher than the value to be reloaded, timeout takes place immediately and no reloading occurs.

NOTE

The value of nominal_time_out (1.4 × ((DFL +2) × 10 bit time)) shown is as per the reset value of LINTOCR[RTO] and LINCR1[CFD]. It changes with change in value of LINTOCR[RTO] and LINCR1[CFD]. Here, 1.4 corresponds to (LINTOCR[RTO])/10 and 2 corresponds to (2- LINCR1[CFD])

Header timeout mechanism

- Master mode: As the header is generated by the LINFlexD, there are only two cases:
  - no error on the bus and timing is correct (nominal header length),
  - an error occurs on the bus and LINFlexD flags it in LINESR register (typically a bit error).

  Therefore there is no meaning of header timeout in master mode, so it is disabled.
- Slave mode
  - header_nominal = 13 + 2 + 10 + 10 = 35 Tbit
  - header_max = 1.4 × Header_nominal = 49 Tbit
  - taking into account a possible 14% clock deviation, header_max seen by LINFlexD is 49 × 1.14 = 56 Tbit
  - If the counter starts after 11 Tbit (break duration), the HTO value is 56 –11 = 45 Tbit.

  The reset value of HTO is 45, and this register can only be programmed in slave mode. Counter restarts after break duration and OC1 gets loaded with the value of HTO.

  As response space is not included in the response, frame timeout is no longer needed and is removed completely. Indeed, header timeout and response timeout cover all cases.

  The timeout counter can be used to detect other timeouts. In this case, the MODE bit must be reset and the output compare value can be updated in the LINTOCR register by software.

Stuck at zero timeout error

If the dominant pulse lasts for a time of at least 100 bits, the SZF bit in LINESR is set. If the same dominant pulse prolongs, the subsequent SZF setting will be 87 bit times apart (instead of 100 bit times).

46.4.2.5.7 Noise

During reception each bit is sampled 16 times and the value of the bit is obtained by taking the majority value of the 8^{th}, 9^{th} and 10^{th} samples. If any one of these three samples has a value different from the other two, this error is flagged.

When OSR = 8, majority of 2nd, 3rd, and 4th samples are taken into account to determine noise. Noise checking is disabled for all OSR less than eight.

This error is flagged when there is noise detected in the start bit (see LIN Error Status Register (LINFlexD_LINESR).

46.4.2.6 Identifier filtering

In LIN protocol the identifier of a message is not associated with the address of a node but is related to the content of the message. A transmitter broadcasts its message to all the receivers. Based on the header received, the receiver decides whether to receive or transmit a response (depending on the identifier value). If the message does not target the node, it should be discarded.

To fulfill this requirement, the LINFlexD provides configurable filters in order to eliminate software intervention. This hardware filtering saves CPU resources that would otherwise be required to perform filtering by software.

There are a maximum of 16 filters (depending on the generic no_of_filters) in the LINFlexD, which can be programmed by the user only during Initialization mode. In order to activate a filter the corresponding FACT bit in IFER needs to be set. There are two modes possible for each identifier depending on the corresponding IFM bit of IFMR.

Identifier list mode

If the n^{th} bit of IFMR is cleared, then filter number 2n and 2n+1 is in identifier list mode. In this mode, the maximum number of filters that can be configured for transmission/ reception equals no_of_filters, depending on the FACT bit of IFER. In this mode the identifier received should match bit by bit to the ID field of IFCR2n or IFCR2n+1 (if the corresponding FACT bit is set).

Identifier mask mode

If the number of filters required is more than no_of_filters, then filters should be configured in Mask mode. If the n^{th} bit of IFMR is set, then filter number 2n is the filter and 2n+1 acts as a mask for it. The FACT bit for filter 2n+1 has no effect in this case. In this mode, if the x^{th} bit of the mask is set, then the x^{th} bit of the received identifier must match the x^{th} bit of filter.

If there is a match of identifier with the mth filter in any mode, then m+1 is loaded in the IFMI register by hardware. No match condition is denoted by IFMI = 0.

Upon matching DFL (2:0), the CCS and DIR bits of the BIDR register are copied from the filter by hardware and from then on, the BIDR register is read-only until the end of the frame. Now if the DIR bit of BIDR is set, then a TXI interrupt is generated if the

HRIE bit of LINIER is set. In this case, software uses the IFMI register to transfer the relevant data from the RAM area to BDR, and after complete transfer the DTRQ bit of LINCR2 is set to start the transmission. If the DIR bit is cleared then an RXI interrupt is generated (provided DRIE bit of LINIER is set) when the checksum has been received and there is no checksum error.

In case of no filter match condition (IFMI = 0) and if the BF bit of LINCR1 is set, then an RXI is generated. Now it is the responsibility of software to configure BIDR and start transmission (by loading the BDR buffer and setting the DTRQ bit of LINCR2) or discard reception (by setting the DDRQ bit of LINCR2). If the BF bit is reset, then the receiver discards the received identifier and turns to Idle state in search of a new break.

Note

If one identifier matches with two filters (one is in the list and the other in Mask mode) then List mode prevails over Mask mode. In mask mode if two filters match with the identifier then the filter having the lower number prevails.

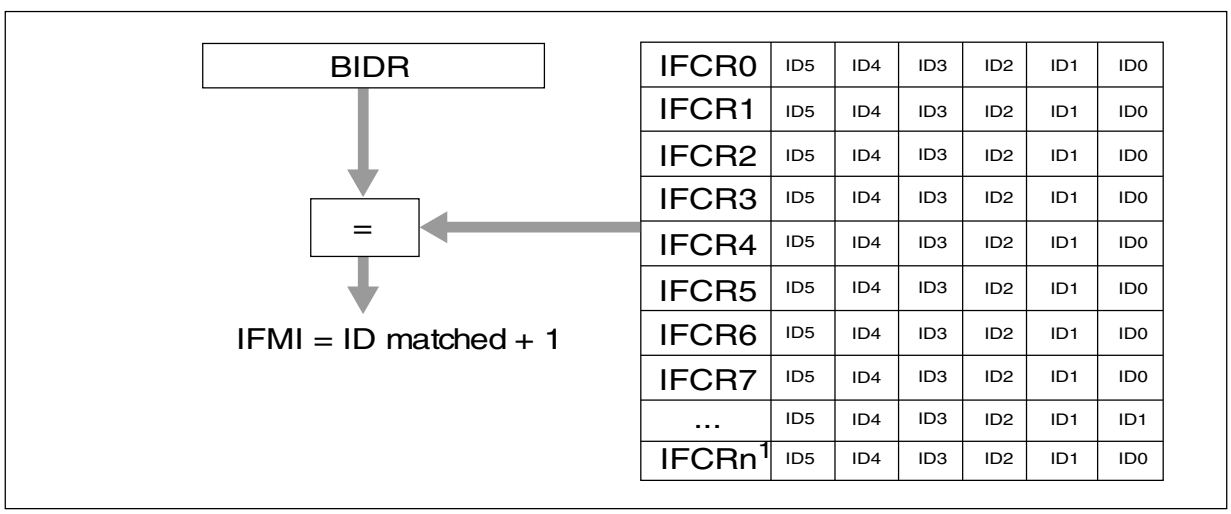


Figure 46-12. Identifier List mode

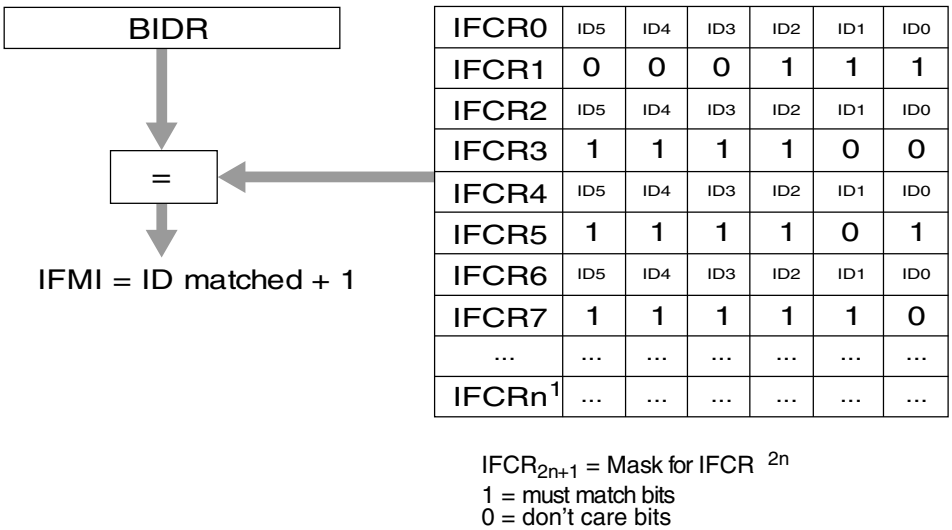


Figure 46-13. Identifier Mask mode

46.4.2.7 Start detection and break delimiter detection in receiver

There is a 10-bit-shift register sample register in the receiver that shifts signal data to the next least significant bit on each incoming sample.

Table 46-3. Start Detection and Delimiter Detection in receiver

9	8	7	6	5	4	3	2	1	0	Sample_reg
—	—	—	—	—	—	—	—	—	RT1	counter = 0
—	—	—	—	—	—	—	—	RT1	RT2	counter = 1
—	—	—	—	—	—	—	RT1	RT2	RT3	counter = 2
—	—	—	—	—	—	RT1	RT2	RT3	RT4	counter = 3
—	—	—	—	—	RT1	RT2	RT3	RT4	RT5	counter = 4
—	—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	counter = 5
—	—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	counter = 6
—	—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	counter = 7
—	RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	counter = 8
RT1	RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	counter = 9
RT2	RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	counter = 10
RT3	RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	counter = 11
RT4	RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	counter = 12
RT5	RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	counter = 13

Table continues on the next page...

Table 46-3. Start Detection and Delimiter Detection in receiver (continued)

9	8	7	6	5	4	3	2	1	0	Sample_reg
RT6	RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	counter = 14
RT7	RT8	RT9	RT10	RT11	RT12	RT13	RT14	RT15	RT16	counter = 15
1	1	1	0	x	0	x	0	x	0	start detect
1	1	1	0	x	0	x	0	x	1	
1	1	1	0	x	0	x	1	x	0	
1	1	1	0	x	1	x	0	x	0	
0	0	0	1	x	1	x	1	x	1	delimiter detect
0	0	0	1	x	1	x	1	x	0	
0	0	0	1	x	1	x	0	x	1	
0	0	0	1	x	0	x	1	x	1	

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

46.4.2.7.1 Start detection mechanism

The following steps are followed for detecting a start bit:

1. Refer to Table 46-3 for the status of the sample register (shift register) when count = 6
2. At this point, if RT1 (the first incoming sample) = 0 and the previous three samples already received are all ones, then it might be a start bit.
3. To make sure it is indeed a start bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority (i.e., 2) out of these 3 samples are equal to ‘0’, then start bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the “verification samples” which are checked at count = 6.

6. The sample register bits 9, 8, 7, and 6 are called “qualification samples” which are checked at count = 6.
7. At count = 9, if majority value of RT8, RT9, and RT10 is not equal to ‘0’ then Noise flag is set.

46.4.2.7.2 Break delimiter detection mechanism

The following steps are followed for detecting a delimiter bit:

1. Refer to Table 46-3 for the status of the sample register (shift register) when count = 6
2. At this point, if RT1 (the first incoming sample) = 1 and the previous three samples already received are all zeros, then it might be a delimiter bit.
3. To make sure it is indeed a delimiter bit, the incoming data samples in the sample register (4), sample register (2) and sample register (0) which correspond to RT3, RT5, and RT7 is verified using the steps mentioned below.
4. If the majority (i.e., 2) out of these 3 samples are equal to ‘1’, then delimiter bit is said to be detected.
5. These three samples RT3, RT5, and RT7 are called the “verification samples” which are checked at count = 6.
6. The sample register bits 9, 8, 7, and 6 are called “qualification samples” which are checked at count = 6.

Hence, a start is detected as soon as it is qualified with 1110 in the sample register and verified with at least two out of three predefined verification samples being zero. Similarly for delimiter detection, qualification samples are 0001 and at least two out of the three verification samples are one. The Noise Flag is set if the start is verified with only two valid samples.

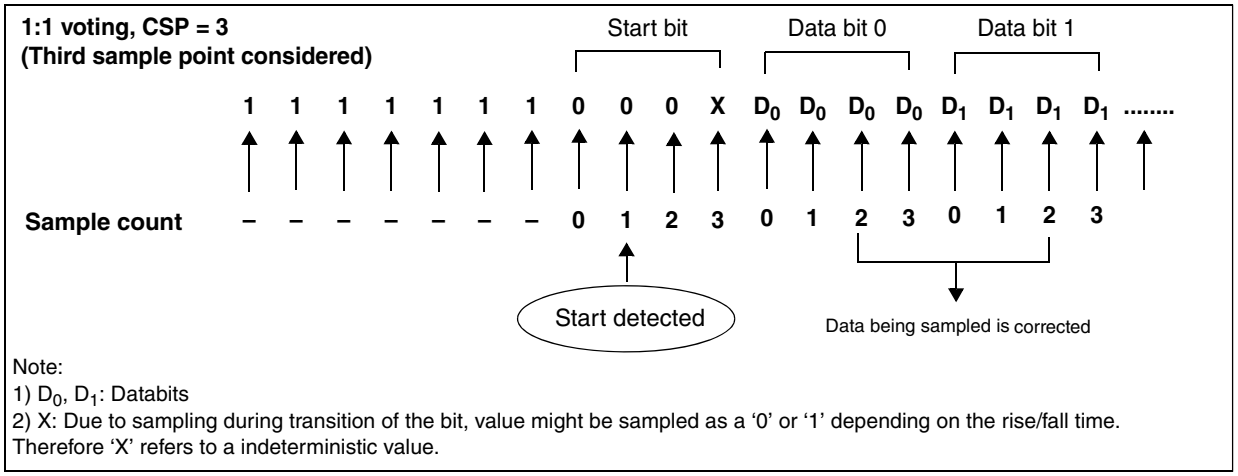


Figure 46-14. Start detection and sampling for over sampling rate = 4 (Case 1)

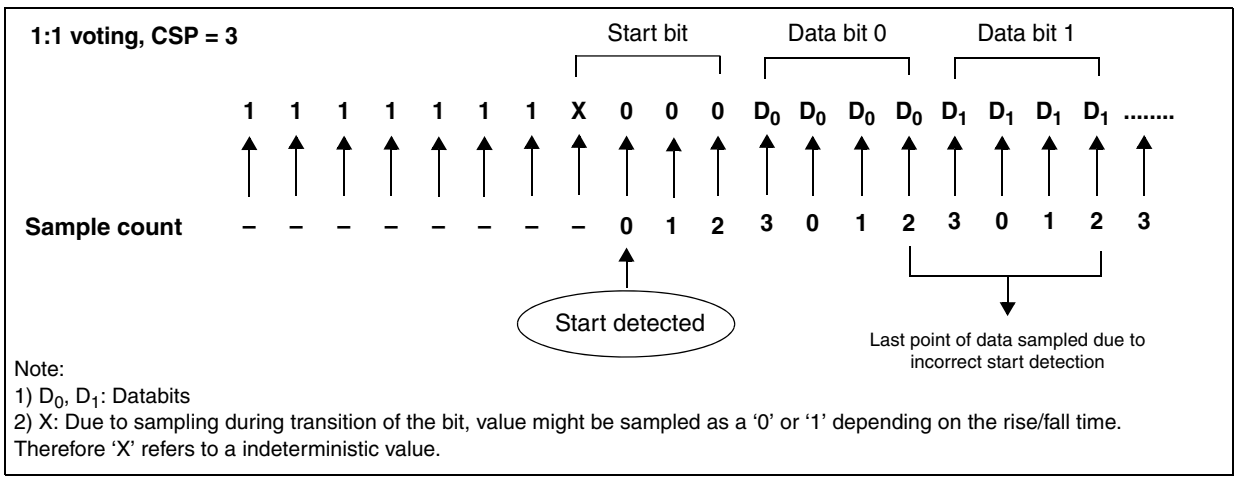


Figure 46-15. Start detection and sampling for over sampling rate = 4 (Case 2a)

The choice of the correct sample points depends on the external Rx signal quality. During the application development process, the error information indicated by the parity error can help to find the best setting for an application-specific hardware signal.

To sample at the correct point therefore, CSP has to be changed to two; resulting in the following situation:

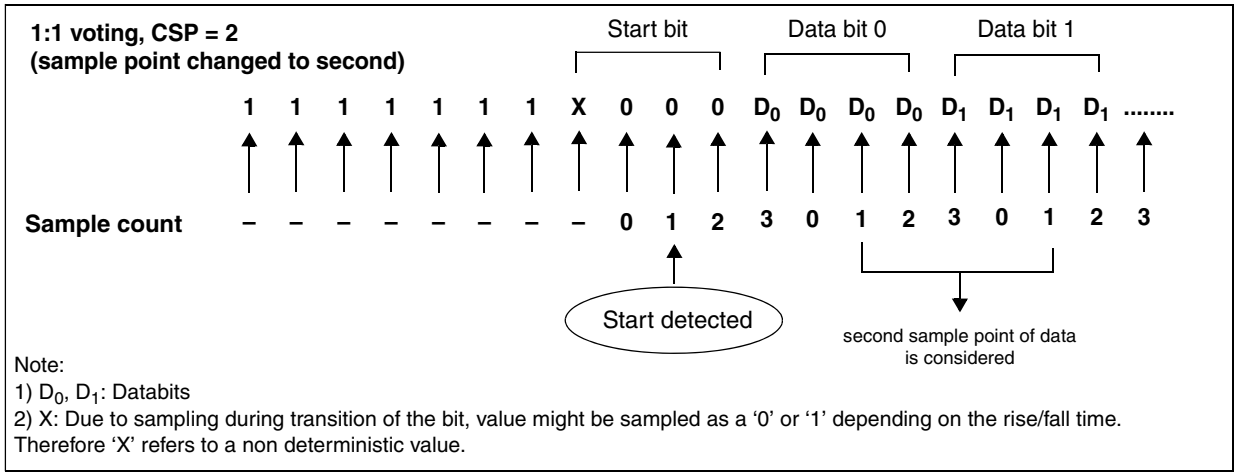


Figure 46-16. Start detection and sampling for over sampling rate = 4 (Case 2b)

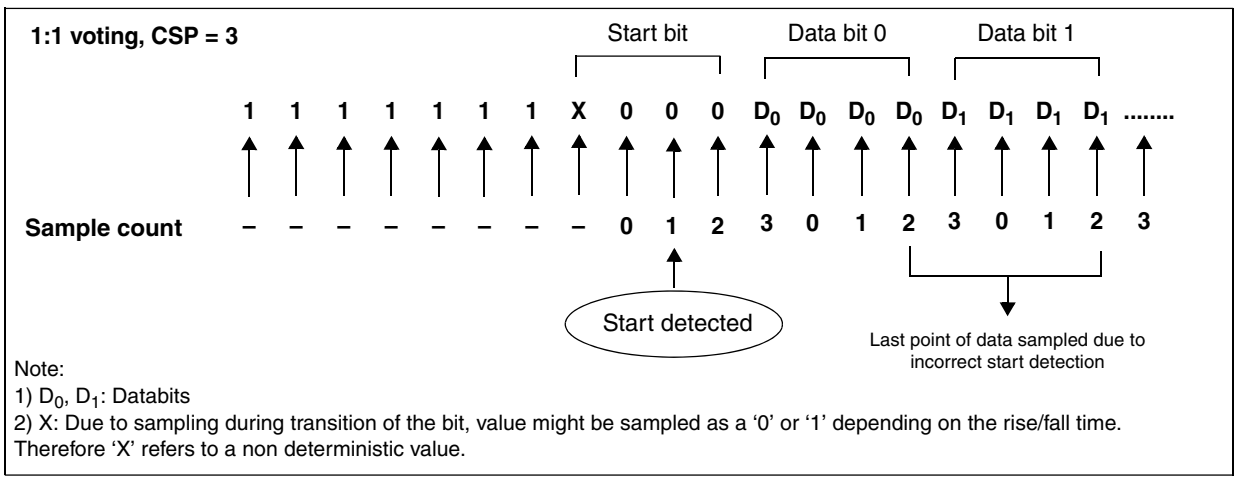


Figure 46-17. Start detection and sampling for over sampling rate = 4 (Case 3)

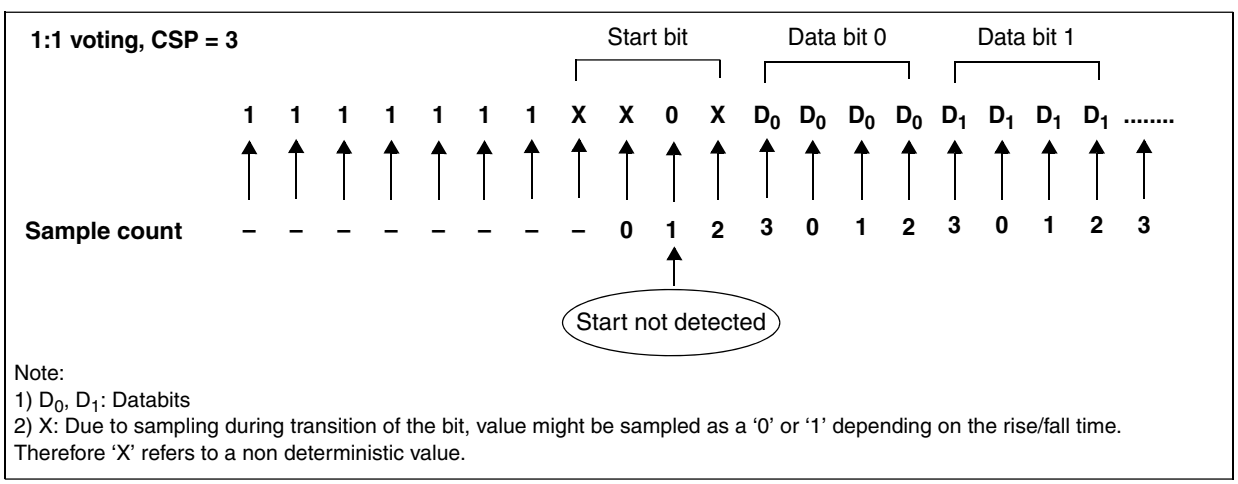


Figure 46-18. Start detection and sampling for over sampling rate = 4 (Case 4)

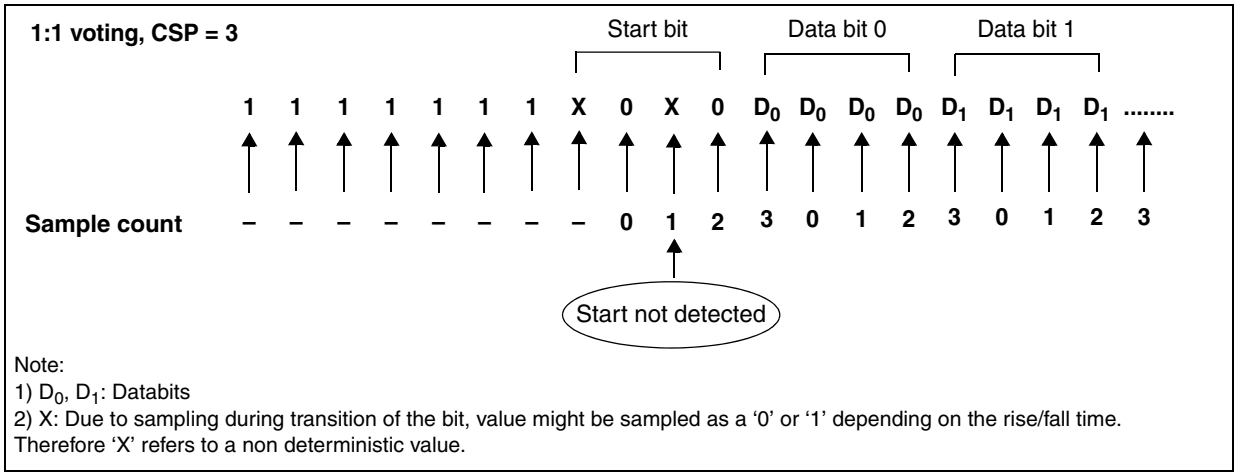


Figure 46-19. Start detection and sampling for over sampling rate = 4 (Case 5)

46.4.2.8 Baud rate generation

The LIN baud rate is programmed in two registers: the LIN Integer Baud Rate Register and the LIN Fraction baud rate register. The Baud Rate Registers can be programmed only during Initialization mode.

Baud rate is calculated with the following formula for both receiver and transmitter.

When ROSE = 1, Tx = Rx = LIN_CLK / (OSR × LDIV). When ROSE = 0, Tx = Rx = LIN_CLK / (16 × LDIV)

Where LIN_CLK is the frequency of the baud clock.

LDIV is an unsigned fixed point number. The mantissa is coded into 20 bits of LINIBRR and the fraction is coded on 4 bits of LINFBRR.

When reduced oversampling is enabled, LINFBRR should not be used and programmed to zero and LDIV contains only the integer part of LINIBRR.

For example: When ROSE = 0 (For LIN and UART mode): LDIV = 468.75 d, LIN_CLK = 36 MHz, LINIBRR = 468 d, LINFBRR = 12 *Baud rate = 36 MHz/ (16 × 468.75) = 4.8 Kbit/s*

For example: When ROSE = 1 (Only for UART mode): LDIV = 10 d, LIN_CLK = 80 MHz, LINIBRR = 4 d, OSR = 4, *Baud rate = LIN_CLK / (OSR × LDIV) = 80 MHz / (4 × 10) = 2 Mbit/s*

46.4.2.9 Automatic resynchronization

To automatically adjust the baud rate based on measurement of the LIN sync field, write the nominal Prescalar value (nominal baud rate) in LINIBRR and LINFBRR, then set the LASE bit in LINCR1 to enable automatic synchronization.

When auto synchronization is enabled, after each LIN Sync Del, the time duration between five falling edges on RDI is sampled on LIN_CLK.

46.4.2.10 Wakeup management

Any node in a sleeping LIN cluster may request a wakeup. The wakeup request is issued by forcing the bus to the dominant state for 250 μs to 5 ms. Every slave node should detect the wakeup request (a dominant pulse longer than 150 μs) and be ready to listen to bus commands within 100 ms, measured from the ending edge of the dominant pulse. The master also wakes on detecting a wakeup request and when the slaves are ready, starts sending frame headers to find out the cause of the wakeup. If the master does not issue a frame header within 150 ms from the wakeup request, then the node issuing a request may try issuing a new wakeup request.

In LINFlexD, a wakeup request can be generated by writing the wakeup character in BDR0 and setting the WURQ bit in LINCR2. On setting the WURQ bit, the character in BDR0 is transmitted. For LIN 2.0, character 0xF0 is sent as the wakeup character.

In LINFlexD, wakeup can be detected in two ways.

1. AUTOWU = 1 On detecting a falling edge in sleep mode, the SLEEP bit is cleared by hardware, the WUF flag is set, and an interrupt is generated if WUPIE is set. LINFlexD is now in normal mode and ready to receive frames.
2. AUTOWU = 0 On detecting a falling edge the WUF flag is set and an interrupt is generated (if WUPIE bit is set). It is then up to the software to clear the SLEEP bit.

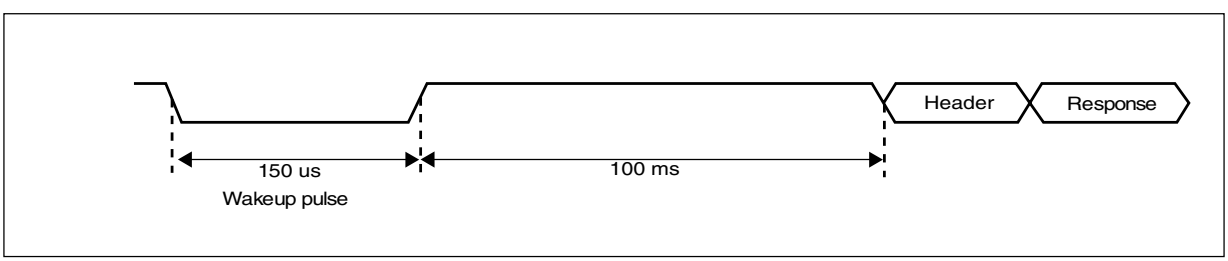


Figure 46-20. Wakeup sequence

46.4.3 Timer

There is an 8-bit counter which has different behavior in different modes as described below:

- **In Output Compare Mode (LINTCSR[MODE]= 1):**

  This counter is running even in Sleep and Initialization mode. The counter value which when matches the two software configurable output compare registers (LINOCR[OC1] or LINOCR[OC2]), generate output compare interrupt (LINESR[OCF]) provided TOCE bit in LINTCSR is set. Once an interrupt occurs subsequent interrupt is generated only when the application has reset the stuck FSM state.
- **In LIN Mode:**

  The software has no control over the TOCE bit and output compare registers are utilized for generation of LIN timeout interrupts (header, frame, response times out). In this case, if LIN moves to Sleep or Init state then this counter remains in Reset state. LIN mode has no meaning if UART is enabled, hence the counter will remain in Reset state.

46.4.4 UART mode

Main features in the UART mode are:

- Full duplex communication
- 8-bit frames, 9-bit frames, 13-bit frames, 16-bit frames, 17-bit frames
- Even/Odd/0/1 Parity
- User programmable over sampling rate to obtain the baud rate of up to 25 Mbit/s

46.4.4.1 8-bit data frames

The eighth bit can be a data or a parity bit. Even/Odd/0/1 parity can be selected by the PC[1:0] bit in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case.

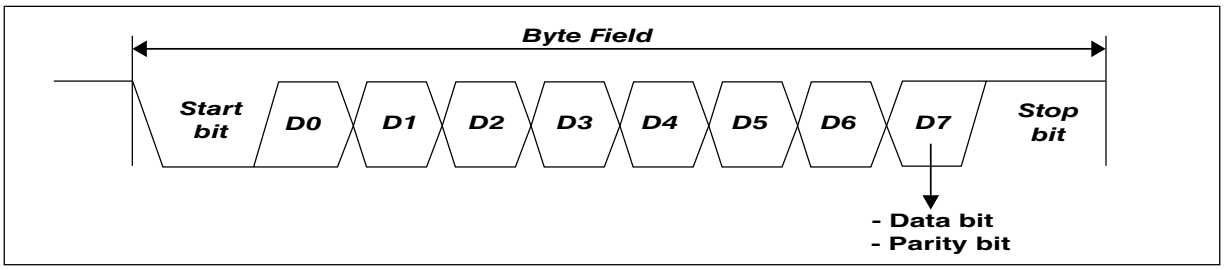


Figure 46-21. UART mode 8-bit data frame

46.4.4.2 9-bit frames

The ninth bit should be a parity bit. Even/Odd/0/1 Parity can be selected by the PC[1:0] field in the same register. An even parity will be set if the modulo-2 sum of the seven data bits is one. An odd parity will be cleared in this case. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

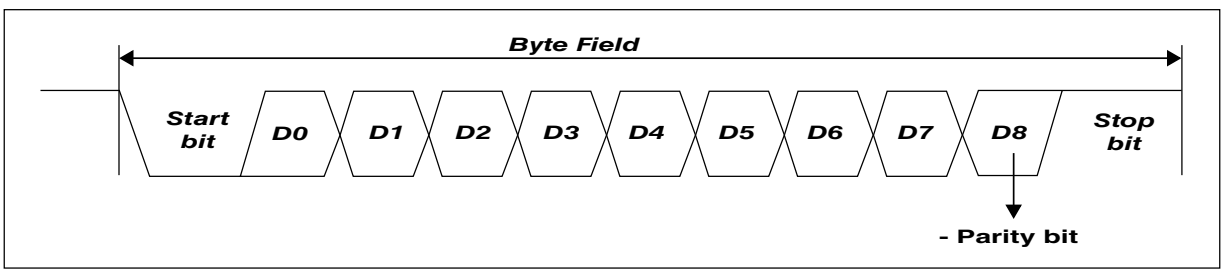


Figure 46-22. UART mode 9-bit data frame

46.4.4.3 16-bit data frames

The sixteenth bit can be a data or a parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

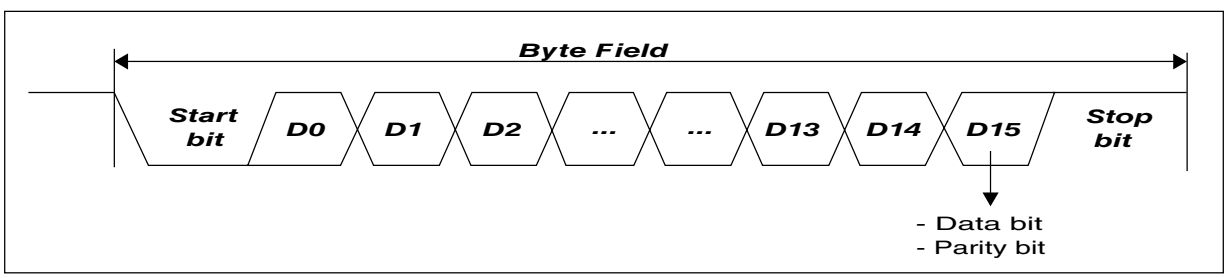


Figure 46-23. UART mode 16-bit data frame

46.4.4.4 17-bit frames

The seventeenth bit is the parity bit. Even/Odd/0/1 Parity bit can be selected by the PC[1:0] bit in the same register. Parity 0 forces a zero logical value. Parity 1 forces a high logical value.

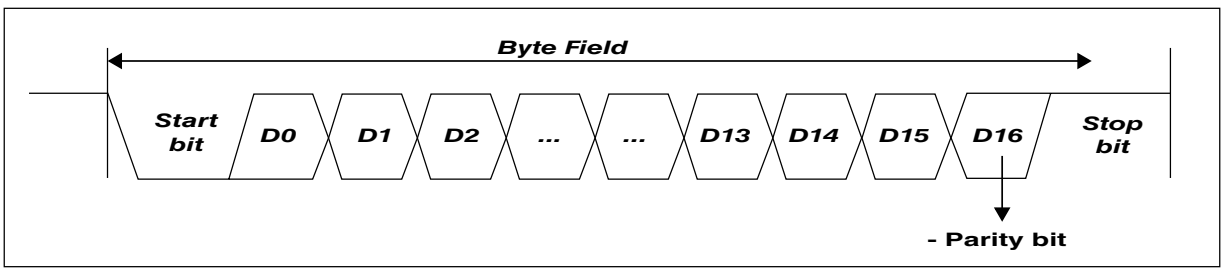


Figure 46-24. UART mode 17-bit data frame

46.4.4.5 13-bit frames

Whenever WLS is one, special word length is selected in UART mode. This bit enables 12-bit + parity bit reception only in FIFO mode.

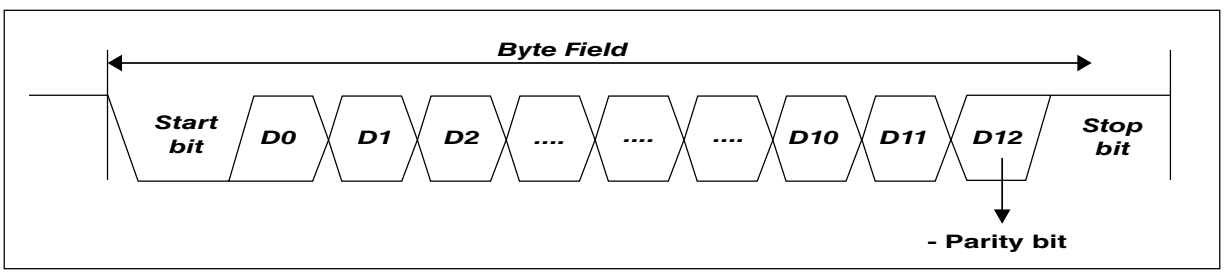


Figure 46-25. UART mode 13-bit data frame

46.4.4.6 Buffer in UART mode

The 8-byte buffer is divided into two parts: one for receiver and one for transmitter, as shown in the following figure:

Tx0	BDR0
Tx1	BDR1
Tx2	BDR2
Tx3	BDR3
Rx0	BDR4
Rx1	BDR5
Rx2	BDR6
Rx3	BDR7

Figure 46-26. Structure of 8-byte buffer

For 16-bit frames, the lower eight bits are written in BDR0 and upper eight bits in BDR1.

The same applies for reception.

46.4.4.7 UART transmitter

UART transmitter

In order to start transmission in the UART mode, UART bit should be set and the transmitter enable bit should be set. Transmission starts when the BDR0 (least significant data byte) is programmed and continues until the number of bytes/halfwords transmitted is equal to the value in the TDFL bits in UARTCR.

The transmit buffer is four bytes (when UARTCR[WL1] = 0) or two halfwords (when UARTCR[WL1] = 1), hence a maximum of four bytes (two halfwords) transmission can be triggered. Once the programmed number of bytes (halfwords) has been transmitted, the DTF flag is set in UARTSR. If the TxEn bit of UART is reset in the middle of a transmission, then the current transmission is completed and no further transmissions can be invoked.

The buffer can be configured in FIFO mode (mandatory when the DMA Tx is enabled) by setting the control bit UARTCR[TFBM].

NOTE

If TFF bit is set and a write is performed to the FIFO, the data transmitted may be erroneous.

Table 46-4. BDRL access in UART mode

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR[WL])	IPS operation result
Write Byte0	BDRL	FIFO	Byte	OK
Write Byte1-2-3	BDRL	FIFO	Byte	IPS transfer error
Write Half-word0-1	BDRL	FIFO	Byte	IPS transfer error
Write Word	BDRL	FIFO	Byte	IPS transfer error
Write Byte0-1-2-3	BDRL	FIFO	Half-word	IPS transfer error
Write Half-word0	BDRL	FIFO	Half-word	OK
Write Half-word1	BDRL	FIFO	Half-word	IPS transfer error
Write Word	BDRL	FIFO	Half-word	IPS transfer error
Read Byte0-1-2-3	BDRL	FIFO	Byte/Half-word	IPS transfer error
Read Half-word0-1	BDRL	FIFO	Byte/Half-word	IPS transfer error
Read Word	BDRL	FIFO	Byte/Half-word	IPS transfer error
Write Byte0-1-2-3	BDRL	BUFFER	Byte/Half-word	OK
Write Half-word0-1	BDRL	BUFFER	Byte/Half-word	OK

Table continues on the next page...

Table 46-4. BDRL access in UART mode (continued)

IPS operation	Register	Mode (UARTCR[TFBM])	Word length (UARTCR[WL])	IPS operation result
Write Word	BDRL	BUFFER	Byte/Half-word	OK
Read Byte0-1-2-3	BDRL	BUFFER	Byte/Half-word	OK
Read Half-word0-1	BDRL	BUFFER	Byte/Half-word	OK
Read Word	BDRL	BUFFER	Byte/Half-word	OK

In UART FIFO mode (UARTCR[TFBM] = 1), any read operation causes an IPS transfer error.

46.4.4.8 UART receiver

UART receiver

The reception of a data byte is started as soon as the user exits initialization mode, sets the RxEn bit, and detects start bit. There is a dedicated 4-byte (if UARTCR[WL1] = 0) or 2-half-words (if UARTCR[WL1] = 1) data buffer for received data. Once the programmed number (RDFL bits) of bytes have been received, the DRF flag is set in UARTSR and the current reception gets completed. RxEn bit needs to be set only to start the reception. The reception is automatically completed as soon as the programmed number (RDFL bits) of bytes have been received.

The buffer can be configured in FIFO mode (mandatory when the DMA Rx is enabled) by setting the control bit UARTCR[RFBM].

Table 46-5. BDRM access in UART mode

IPS operation	Register	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read Byte4	BDRM	FIFO	Byte	OK
Read Byte5-6-7	BDRM	FIFO	Byte	IPS transfer error
Read Half-word2-3	BDRM	FIFO	Byte	IPS transfer error
Read Word	BDRM	FIFO	Byte	IPS transfer error
Read Byte4-5-6-7	BDRM	FIFO	Half-word	IPS transfer error
Read Half-word2	BDRM	FIFO	Half-word	OK
Read Half-word3	BDRM	FIFO	Half-word	IPS transfer error
Read Word	BDRM	FIFO	Half-word	IPS transfer error
Write Byte4-5-6-7	BDRM	FIFO	Byte/Half-word	IPS transfer error
Write Half-word2-3	BDRM	FIFO	Byte/Half-word	IPS transfer error
Write Word	BDRM	FIFO	Byte/Half-word	IPS transfer error
Read Byte4-5-6-7	BDRM	BUFFER	Byte/Half-word	OK

Table continues on the next page...

Table 46-5. BDRM access in UART mode (continued)

IPS operation	Register	Mode (UARTCR[RFBM])	Word length (UARTCR:WL)	IPS operation result
Read Half-word2-3	BDRM	BUFFER	Byte/Half-word	OK
Read Word	BDRM	BUFFER	Byte/Half-word	OK
Write Byte4-5-6-7	BDRM	BUFFER	Byte/Half-word	IPS transfer error
Write Half-word2-3	BDRM	BUFFER	Byte/Half-word	IPS transfer error
Write Word	BDRM	BUFFER	Byte/Half-word	IPS transfer error

Note

Refer to the layout of the registers BDRL and BDRM to identify the mapping between byte x and data bits of the registers BDRL/BDRM.

Note

- If the user does not know in advance how many bytes are to be received, RDFL should not be programmed in advance. The reset value of RDFL is zero. This will ensure that the reception will happen byte by byte. The state machine will move to the Idle state after each byte reception.
- If RDFL is programmed for a certain value but that number of bytes are not received, then reception will hang. In that case, software needs to take care of timeout by seeing the flag. Software has to set the sleep bit to move to Idle state.
- If a STOP request arrives in the middle of one reception, it is only acknowledged after all the programmed number of data bytes have received; it is not served immediately. If the programmed number of data bytes are not received, then software has to take care of timeout. When the state machine moves to Idle state, then only a stop request is served.
- If during reception of any byte a parity error occurs, then the corresponding PEx bit in UARTSR is set. No interrupt is generated in this case. If a framing error occurs in any byte (FE bit in UARTSR is set) then an interrupt is generated if FEIE bit in LINIER is set. As there is only one register bit for framing error, this interrupt will be helpful in identifying which byte has framing error.

- If the last received frame has not been read from the buffer (in other words, RMB is not reset by the user), then upon reception of the next byte, an overrun error occurs (BOF bit in UARTSR will be set) and one message is depending on RBLM bit of LINCR1. An interrupt is generated if the BOIE bit in LINIER is set.
- When WLS bit = 1, BDRM access depends on WL bit setting.
- When WLS bit = 1, WL = 0 or 1 should not be used, since this will lead to incorrect reception of data.
- When IPG_STOP is requested, for UART in FIFO mode, SW has to set the INIT bit in order to send the receiver to idle. This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

46.4.5 DMA interface

46.4.5.1 Main features

The LINFlexD DMA offers a parametric and programmable solution with the following distinctive features:

- LIN Master node, TX mode: single DMA channel
- LIN Master node, RX mode: single DMA channel
- LIN Slave node, TX mode: 1 to N DMA channel where N = max number of ID filters
- LIN Slave node, RX mode: 1 to N DMA channel where N = max number of ID filters
- UART node, TX mode: single DMA channel
- UART node, RX mode: single DMA channel + time-out

46.4.5.2 Definitions

Control/status register fields of the LINFlexD macrocell are described in Table 46-6 and Table 46-7.

Table 46-6. LINFlexD control/status fields description

Register	Field	Level	Description
LINCR2	DDRQ	High	Data discard request in reception mode
	DTRQ	High	Data transmission request (slave mode) of the LIN data field stored in BDRL/BDRM
	HTRQ	High	Header transmission request (master mode)
BIDR	DFL	—	Data field length: • 1 to 8 bytes: normal frames
	DIR	—	Direction of the data field: • 0 => reception • 1 => transmission
	CCS	—	Checksum type: • 0 enhanced • 1 classic
	ID[5:0]	—	Identifier.
LINSR	RMB	—	Buffer data ready to be read via CPU/DMA. • 0 buffer data is free • 1 buffer data is ready
	DRF	High	Data reception completed
	DTF	High	Data transmission completed
	HRF	High	Header received. Used in slave mode in case of Tx filter match.
IFMI	IFMIx	!= zero	Filter match index (slave mode).
UARTCR	FBM	—	FIFO/Buffer mode • 0 Buffer mode • 1 FIFO mode (mandatory in DMA mode)
UARTSR	RFE	—	Rx FIFO empty • 0 Rx FIFO not empty • 1 Rx FIFO empty
	TFF	—	Tx FIFO full • 0 Tx FIFO not full • 1 Tx FIFO full
DMATXE[2**TX_CH_NUM][1]		High	DMA Tx channel enable (read/write). In UART or LIN master mode only the channel 0 can be programmed. In LIN slave mode: # DMA TX channel = IFMI[3:0] -1.

Table continues on the next page...

Table 46-6. LINFlexD control/status fields description (continued)

Register	Field	Level	Description
			DMATXE[x] = 0b => TX channel x disabled DMATXE[x] = 1b => TX channel x enabled

DMARXE[2**RX_CH_NUM][1]		High	DMA Rx channel enable (read/write). In UART or LIN master mode only the channel 0 can be programmed.
			In LIN slave mode: # DMA RX channel = IFMI[3:0] -1.
			DMARXE[x] = 0b => RX channel x disabled
			DMARXE[x] = 1b => RX channel x enabled

1. ** stands for exponentiation.
1. ** stands for exponentiation

Table 46-7. LINFlexD DMA control fields description

Field	Mode	Value	Level	Description
DMA_TEN	LIN master Tx or UART Tx	DMATXE[0]	High	Logical AND between the 2 DMA enable bits (LINFlexD and eDMA).
DMA_TEN	LIN slave Tx	DMATXE[x] x = IFMI — 1	High	
DMA_REN	LIN master Rx or UART Rx	DMARXE[0]	High	
DMA_REN	LIN slave Rx	DMARXE[x] x = IFMI — 1	High	

Control/status fields of the TCD descriptors referred to in this document are described in Table 46-8.

Table 46-8. TCD control fields description

TCD Field	Level	Description
CITER[14:0]	—	Current "major" iteration count
BITER[14:0]	—	Beginning "major" iteration count
NBYTES[31:0]	—	Inner "minor" byte transfer count. Number of bytes to be transferred in each service request of the channel.
SADDR[31:0]	—	Source address
SOFF[15:0]	—	Source address signed offset applied to the current source address as each source read is completed.
SSIZE[2:0]	—	Source data transfer size 000 => 8-bit 001 => 16-bit

Table continues on the next page...

Table 46-8. TCD control fields description (continued)

TCD Field	Level	Description
		010 => 32-bit 011 => 64-bit
SLAST[31:0]	—	Last source address adjustment
DADDR[31:0]	—	Destination address
DOFF[15:0]	—	Destination address signed offset applied to the current destination address as each destination write is completed.
DSIZE[2:0]	—	Destination data transfer size 000 => 8-bit 001 => 16-bit 010 => 32-bit 011 => 64-bit
DLAST_SGA[31:0]	—	Last destination address adjustment or the memory address for the next TCD to be loaded into this channel (scatter/gather)
INT_MAJ	High	Enable an interrupt when major iteration count completes
START	High	The channel is explicitly started via a software initiated service request.
DONE	High	Channel done (the DMA has completed the outer major loop).
D_REQ	High	Disable request. If this flag is set the DMA hardware automatically clears the corresponding DMAERQ bit when the current major iteration count reaches zero.

46.4.5.3 Master node –TX mode

On a master node, in TX mode the DMA interface requires a single TX channel. Each TCD controls a single frame. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

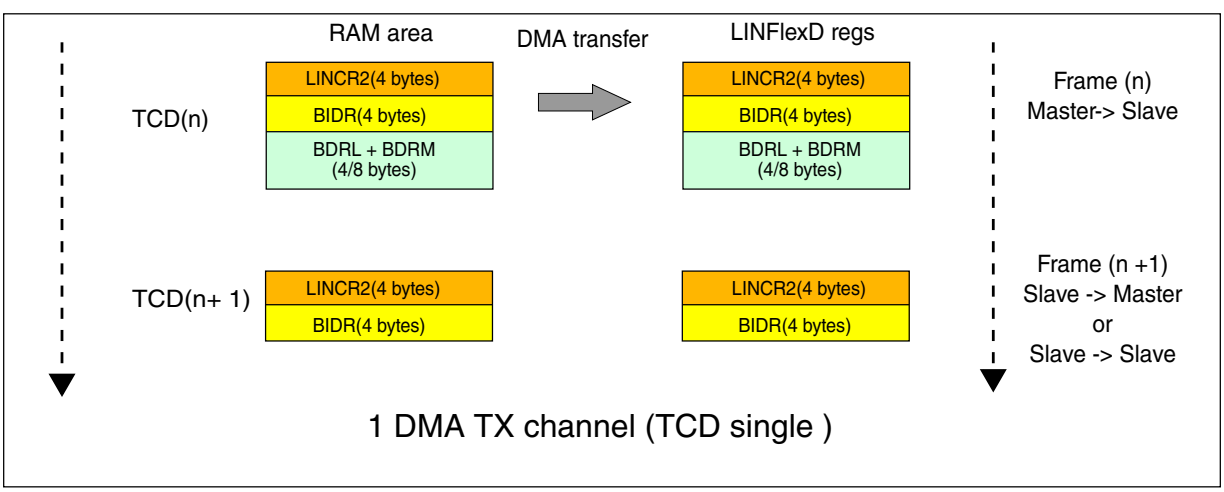


Figure 46-27. Master node –TX memory map

The TCD chain of the DMA Tx channel on a master node supports:

- Master to Slave: transmission of the entire frame (header + data)
- Slave to Master: transmission of the header; the data reception is controlled by the Rx channel on the master node
- Slave to Slave: transmission of the header

The following table provides the register settings of the LINCR2 and BIDR for each class of LIN frame.

Table 46-9. Master node – Tx mode – Register setting

LIN frame	LINCR2	BIDR
Master to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)
Slave to Master	DDRQ = 0 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)
Slave to Slave	DDRQ = 1 DTRQ = 0 HTRQ = 0	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Tx interface is given in the following figure. DMA TX FSM moves to Idle state immediately at next clock edge if DMATXE[0] = 0. The TCD setting (word transfer) is given in Table 46-10. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 46-10. TCD setting – Master node – Tx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4 + 4] + 0/4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. LINCR2 + BIDR + BDRL + BDRM
SADDR[31:0]	-	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer

Table continues on the next page...

Table 46-10. TCD setting – Master node – Tx mode (continued)

TCD Field	Value	Description
SLAST[31:0]	–N	
DADDR[31:0]	-	LINCR2 address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No SW request

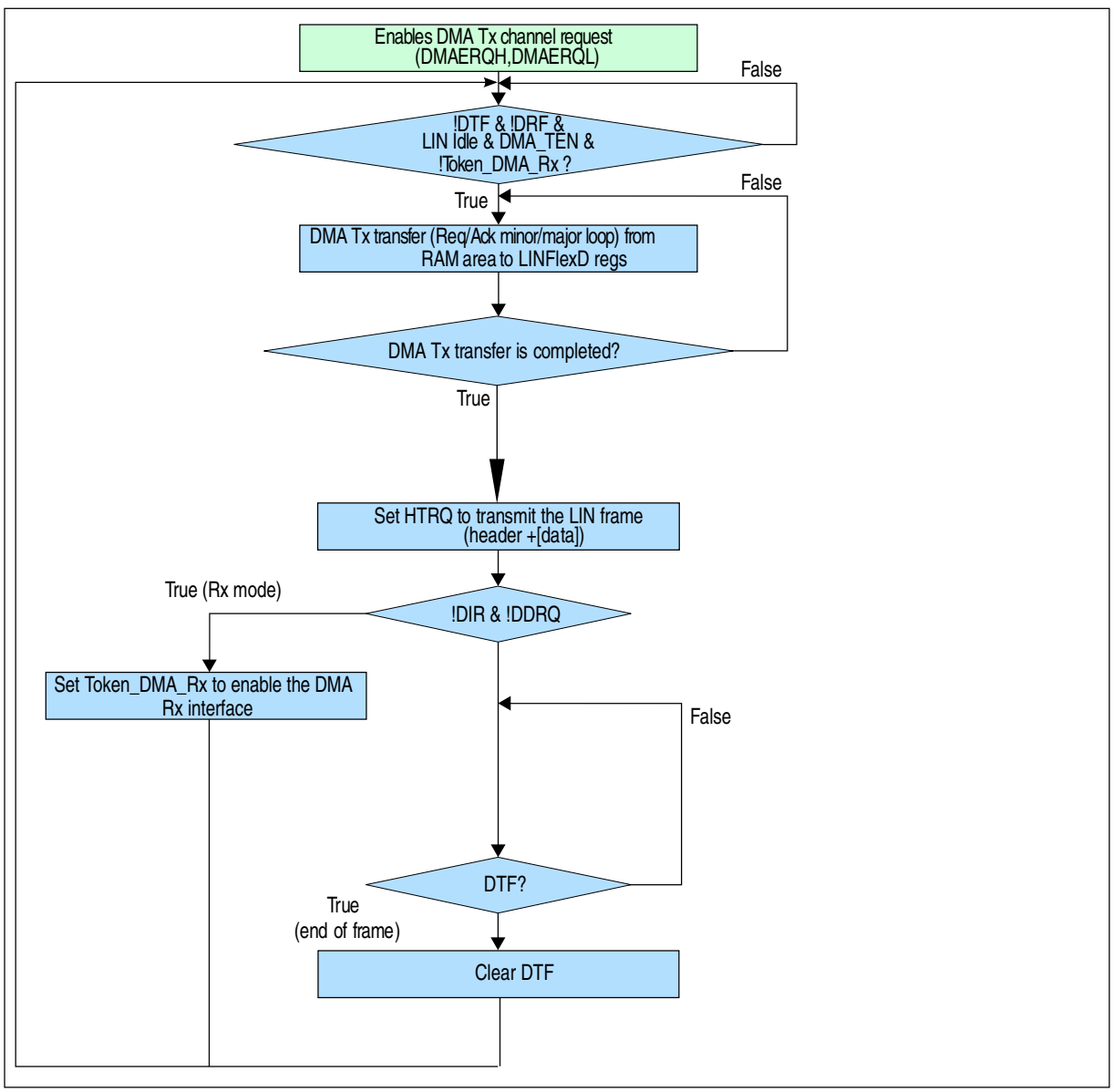


Figure 46-28. Master node – DMA Tx FSM (concept scheme)

46.4.5.4 Master node - RX mode

On a master node in RX mode, the DMA interface requires a single RX channel. Each TCD controls a single frame . The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

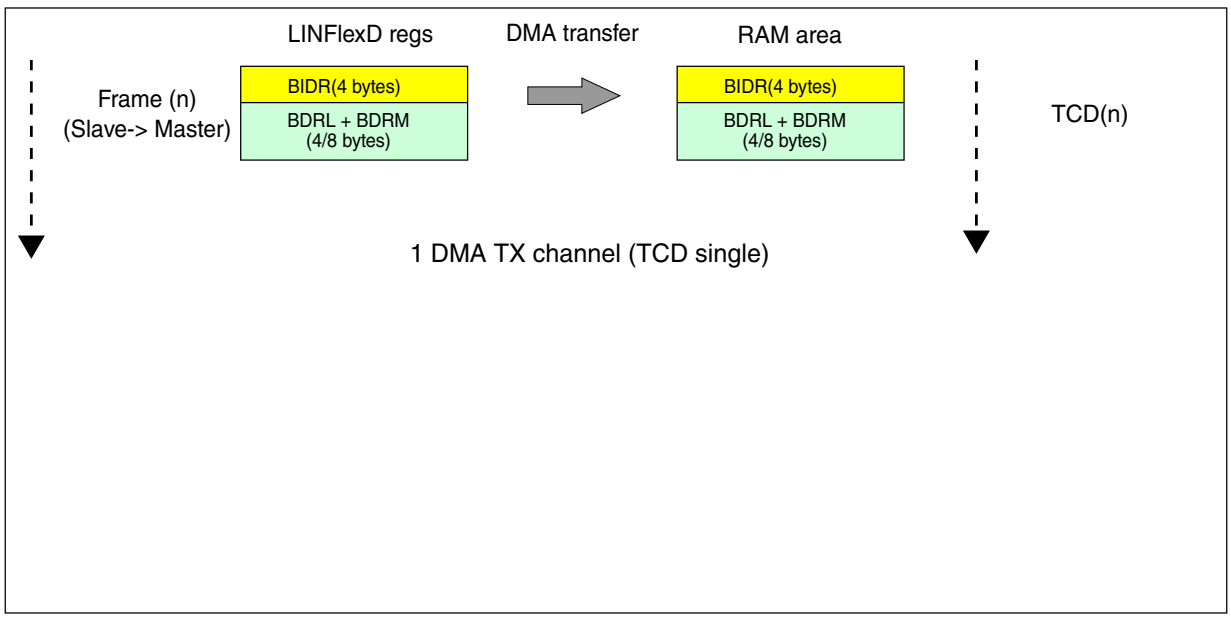


Figure 46-29. Master node – RX memory map

The TCD chain of the DMA Rx channel on a master node supports Slave to Master: reception of the data field of the header

The BIDR register is optionally copied into the RAM area. This BIDR field (part of FIFO data) contains the ID of each message, which only allows the CPU to figure out which ID the LINFlexD DMA received if the single DMA channel setup is used.

The concept FSM to control the DMA Rx interface is given in the following figure. DMA RX FSM moves to Idle state immediately at the next clock edge if DMARXE[0] = 0. The TCD setting (word transfer) is given in the next table. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 46-11. TCD setting – Master node – Rx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4] +4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BIDR + BDRL + BDRM

Table continues on the next page...

Table 46-11. TCD setting – Master node – Rx mode (continued)

TCD Field	Value	Description
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain
START	0	No SW request

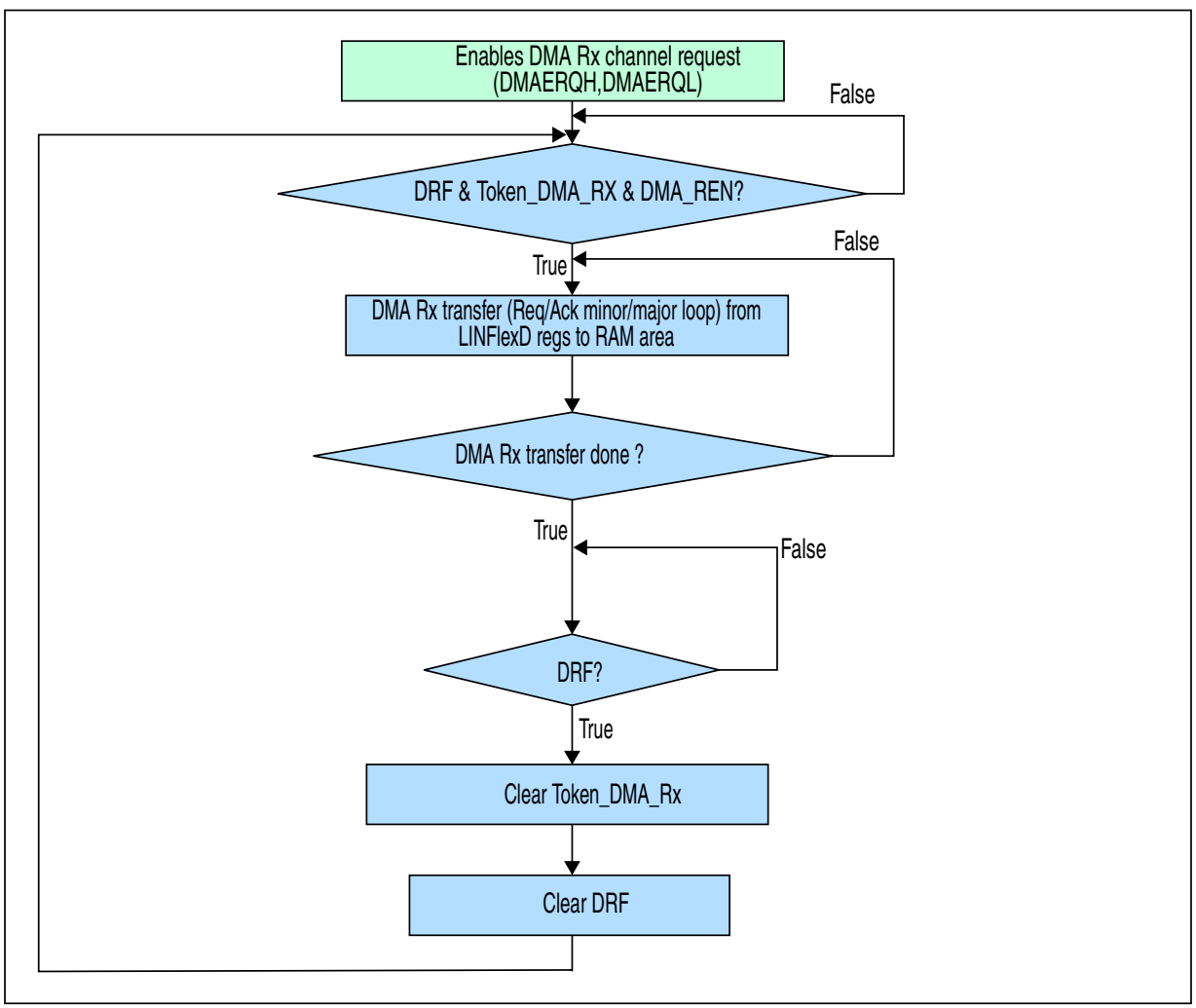


Figure 46-30. Master node – DMA Rx FSM (concept scheme)

46.4.5.5 Slave node – TX mode

On a slave node in TX mode, the DMA interface requires a DMA TX channel for each ID filter programmed in TX mode. In case a single DMA TX channel is available, a single ID field filter must be programmed in TX mode. Each TCD controls a single frame. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

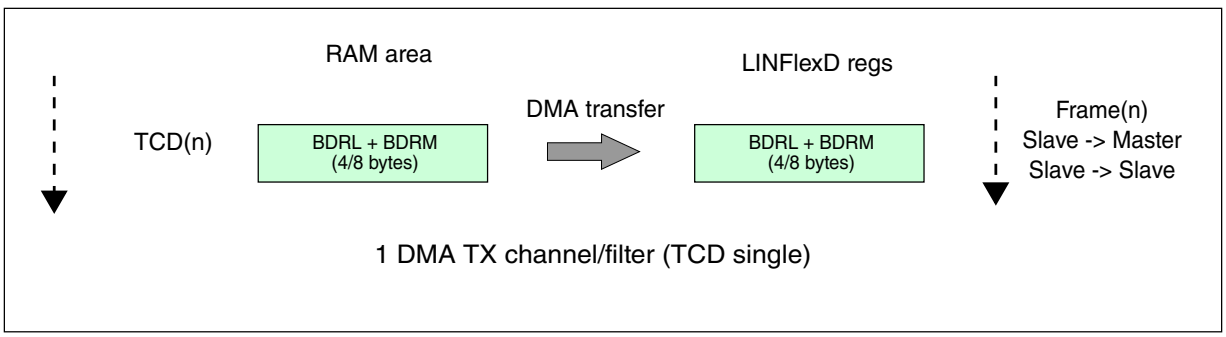


Figure 46-31. Slave node – TX memory map

The TCD chain of the DMA Tx channel on a slave node supports:

- Slave to Master: transmission of the data field
- Slave to Slave: transmission of the data field

The register setting of the LINCR2, IFER, IFMR, and IFCR registers is given in Table 46-12.

Table 46-12. Slave node – Tx mode – Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR
Slave to Master or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Tx mode) for each DMA TX channel	• Identifier list mode • Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 1 (TX)

The concept FSM to control the DMA Tx interface is given in the following figure. DMA TX FSM moves to Idle state if DMATXE[x] = 0 where x = IFMI – 1.The TCD setting (word transfer) is given in Table 46-13. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 46-13. TCD setting – Slave node – Tx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BDRL + BDRM
SADDR[31:0]	—	RAM address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—
DADDR[31:0]	—	BDRL address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain.
START	0	No SW request

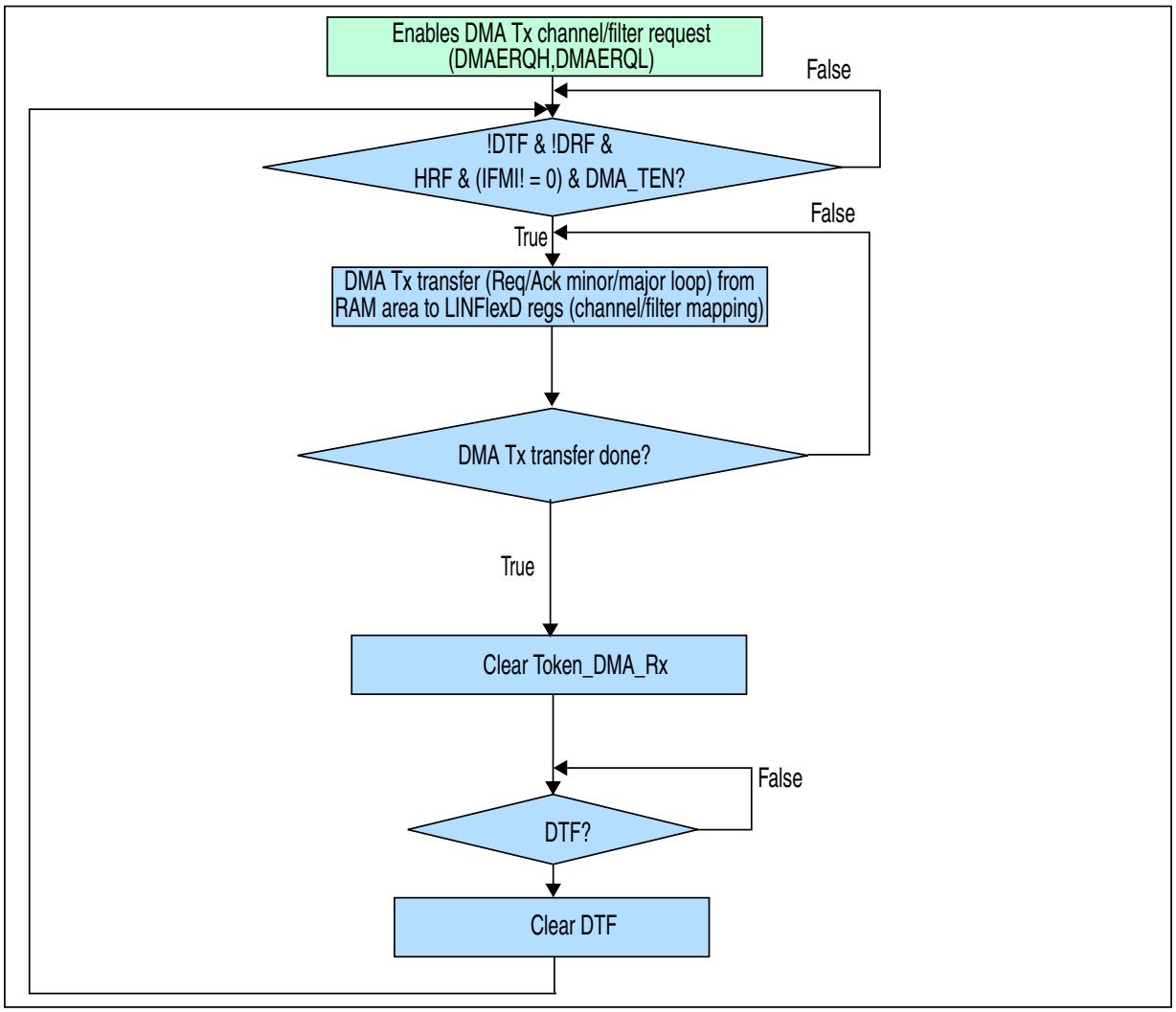


Figure 46-32. Slave node – DMA Tx FSM (concept scheme)

46.4.5.6 Slave node – RX mode

On a slave node in RX mode, the DMA interface requires a DMA RX channel for each ID filter programmed in RX mode. In case a single DMA RX channel is available, a single ID field filter must be programmed in RX mode. Each TCD controls a single frame. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in the following figure.

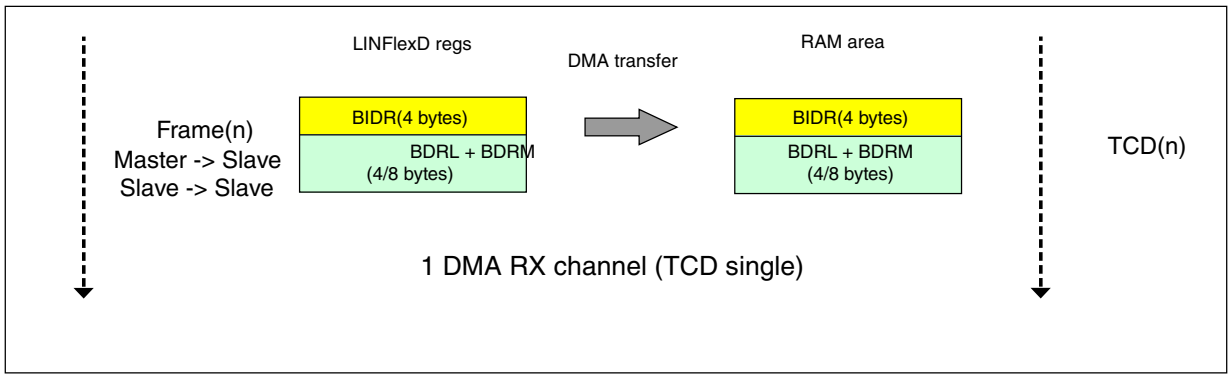


Figure 46-33. Slave node – RX memory map

The TCD chain of the DMA Rx channel on a slave node supports:

- Master to Slave: reception of the data field
- Slave to Slave: reception of the data field

The register setting of the LINCR2, IFER, IFMR, IFCR registers is given in Table 46-14.

Table 46-14. Slave node – Rx mode – Register setting

LIN frame	LINCR2	IFER	IFMR	IFCR
Master to Slave or Slave to Slave	DDRQ = 0 DTRQ = 0 HTRQ = 0	To enable an ID filter (Rx mode) for each DMA RX channel	• Identifier list mode • Identifier mask mode	DFL = payload size ID = address CCS = checksum DIR = 0 (RX)

The concept FSM to control the DMA Rx interface is given in the following figure. DMA Rx FSM moves to Idle state if DMARXE[x] = 0 where x = IFMI – 1. The TCD setting (word transfer) is given in Table 46-15. All other TCD fields = 0. TCD settings based on halfword or byte transfer are allowed.

Table 46-15. TCD setting – Slave node – Rx mode

TCD Field	Value	Description
CITER[14:0]	1	Single iteration for the "major" loop
BITER[14:0]	1	Single iteration for the "major" loop
NBYTES[31:0]	[4] + 4/8 = N	Data buffer is stuffed with dummy bytes if the length is not word aligned. BIDR + BDRL + BDRM
SADDR[31:0]	—	BIDR address
SOFF[15:0]	4	Word increment
SSIZE[2:0]	2	Word transfer
SLAST[31:0]	–N	—

Table continues on the next page...

Table 46-15. TCD setting – Slave node – Rx mode (continued)

TCD Field	Value	Description
DADDR[31:0]	—	RAM address
DOFF[15:0]	4	Word increment
DSIZE[2:0]	2	Word transfer
DLAST_SGA[31:0]	–N	No scatter/gather processing
INT_MAJ	0/1	Interrupt disabled/enabled
D_REQ	1	Only on the last TCD of the chain.
START	0	No SW request

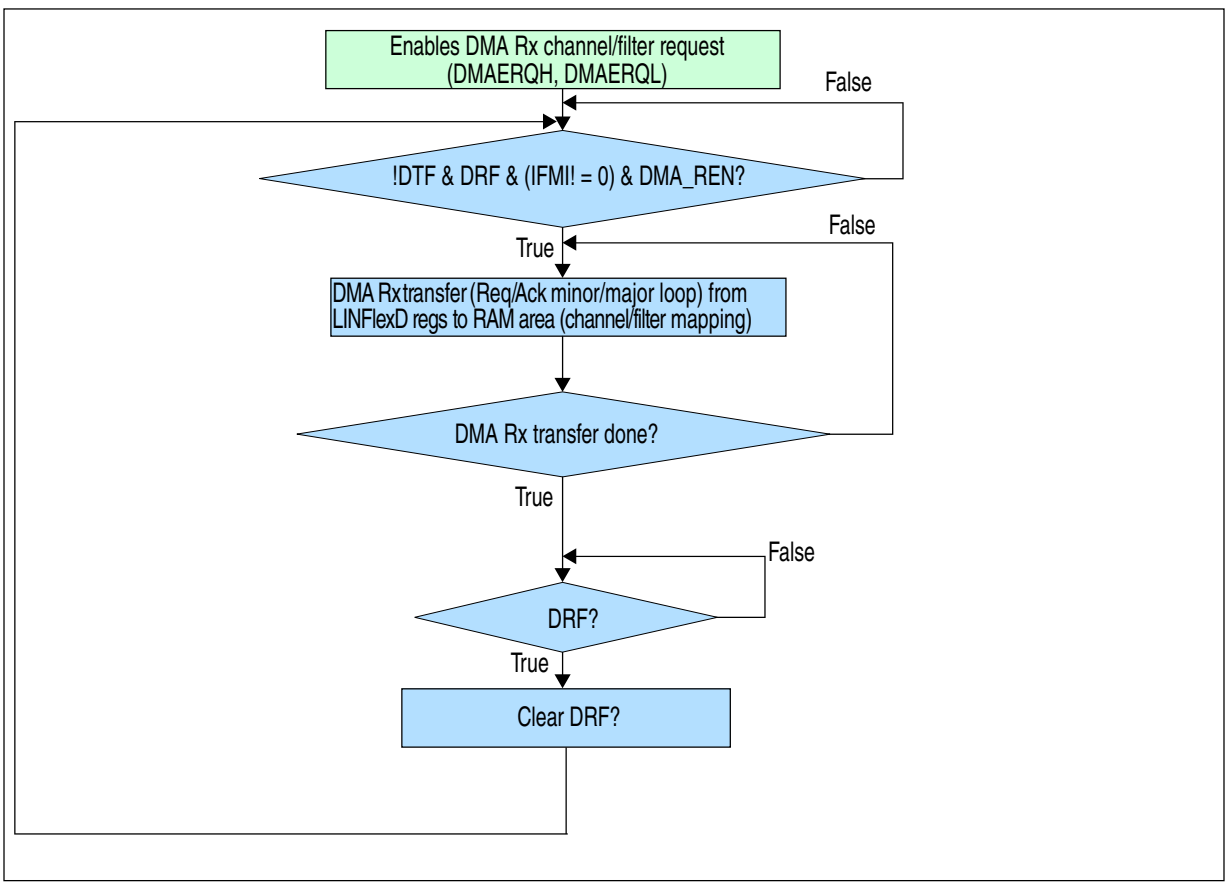


Figure 46-34. Slave node – DMA Rx FSM (concept scheme)

46.4.5.7 UART – TX mode

In UART TX mode, the DMA interface requires a DMA TX channel. A single TCD can control the transmission of an entire Tx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in Figure 46-35.

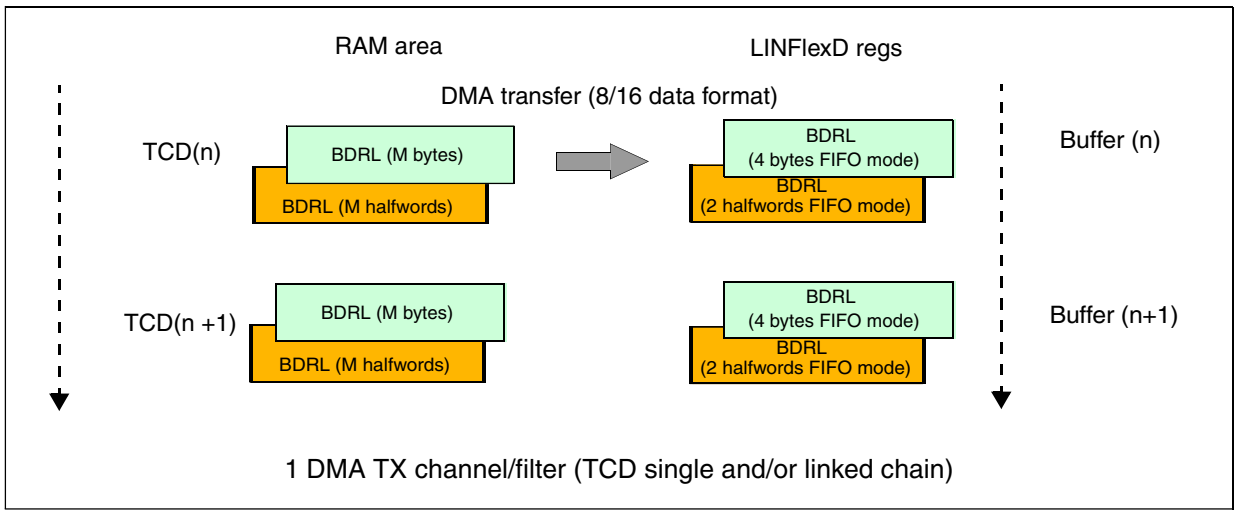


Figure 46-35. UART – TX memory map

The UART TX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Absorb the latency, following a DMA request (due to the DMA arbitration), to move data from the RAM to the FIFO
- Use low priority DMA channels

The Tx FIFO size is:

- 4 bytes in 8 bit data format
- 2 halfwords in 16-bit data format

A DMA request is triggered by FIFO-not-full (TX) status signals.

The concept FSM to control the DMA Tx interface is given in Figure 46-36. DMA Tx FSM will move to Idle state if DMATXE[0] = 0.The TCD setting (typical case) is given in Table 46-16. All other TCD fields = 0. The minor loop transfers a single byte/halfword as soon as a free entry is available in the Tx FIFO.

Table 46-16. TCD setting – UART – Tx mode

TCD Field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the "major" loop
BITER[14:0]	M		Multiple iterations for the "major" loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	-		RAM address

Table continues on the next page...

Table 46-16. TCD setting – UART – Tx mode (continued)

TCD Field	Value		Description
	8 bits data	16 bits data	
SOFF[15:0]	1	2	Byte/Half-word increment
SSIZE[2:0]	0	1	Byte/Half-word transfer
SLAST[31:0]	–M	–M × 2	-
DADDR[31:0]	-		BDRL address
DOFF[15:0]	0		No increment (FIFO)
DSIZE[2:0]	0	1	Byte/Half-word transfer
DLAST_SGA[31:0]	0		No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain.
START	0		No SW request

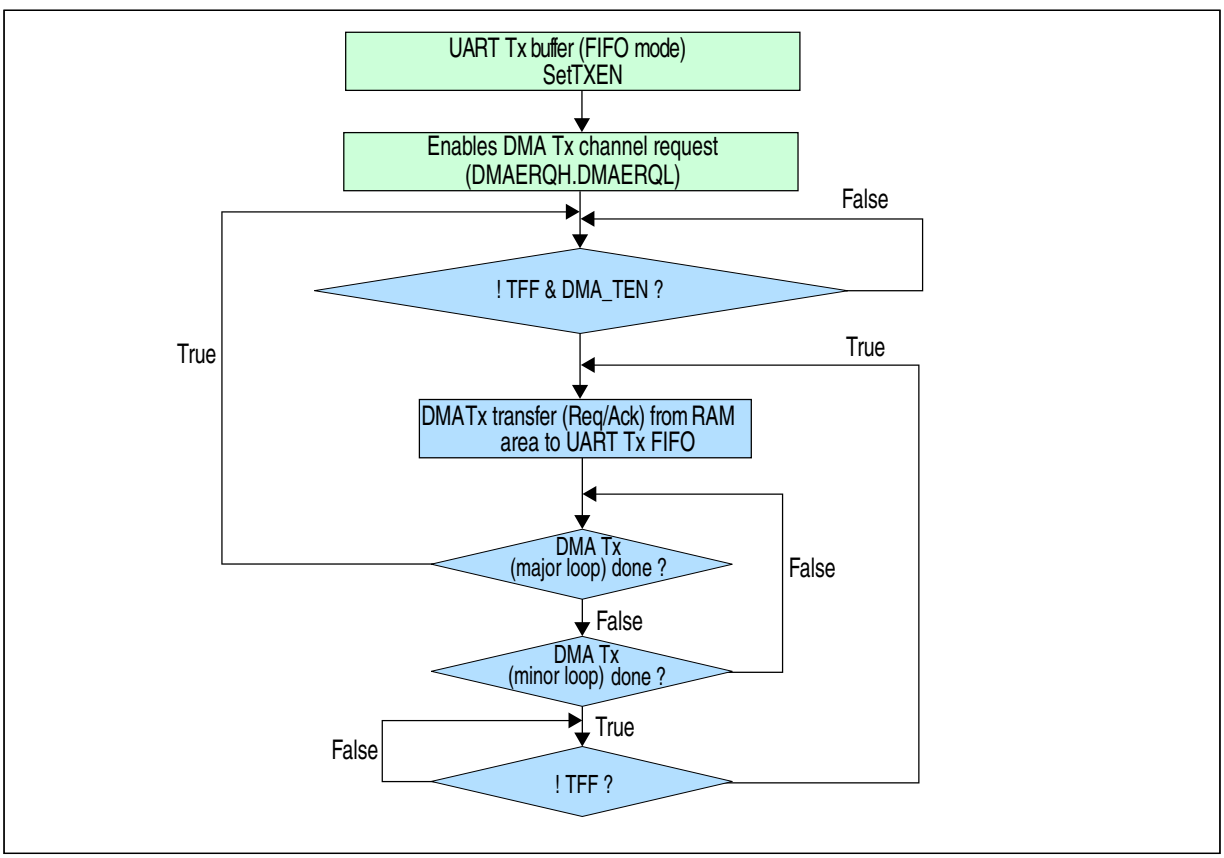


Figure 46-36. UART – DMA Tx FSM (concept scheme)

46.4.5.8 UART – RX mode

In UART RX mode, the DMA interface requires a DMA RX channel. A single TCD can control the reception of an entire Rx buffer. The memory map associated with the TCD chain (RAM area and LINFlexD registers) is given in Table 46-17.

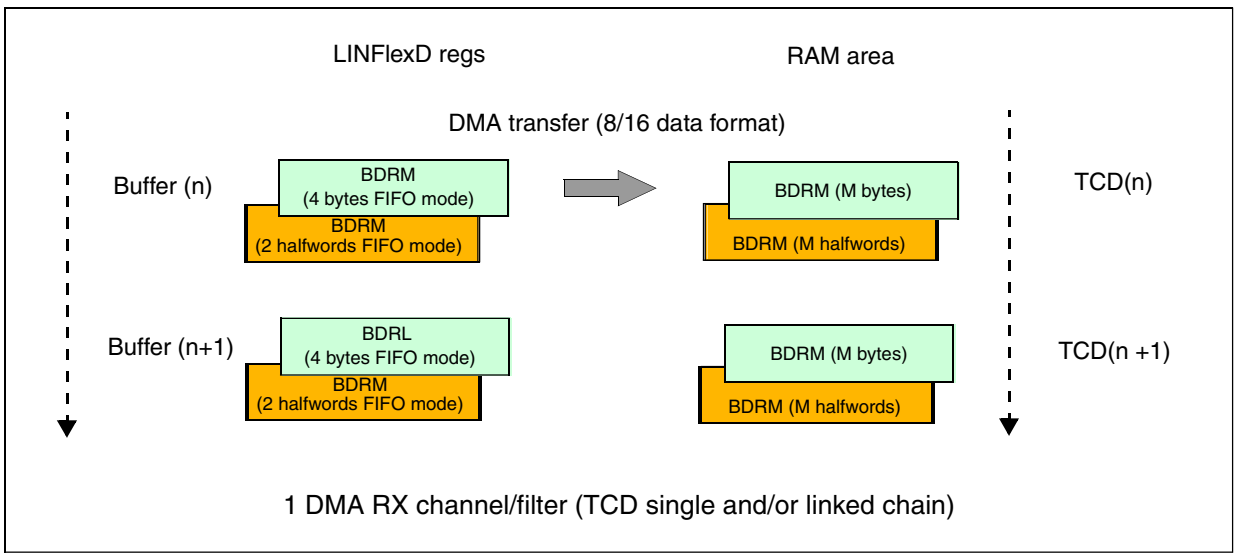


Figure 46-37. UART – RX memory map

The UART RX buffer must be configured in FIFO mode in order to:

- Allow the transfer of large data buffer by a single TCD
- Absorb the latency, following a DMA request (due to the DMA arbitration), to move data from the FIFO to the RAM
- Use low priority DMA channels

The Rx FIFO size is:

- 4 bytes in 8-bit data format

This will sufficient because just one byte allows a reaction time of about 3.8 µs (at 2 Mbit/s) (~450 clock cycles at 120 MHz) before the transmission is affected. A DMA request is triggered by FIFO and not by empty (Rx) status signals.

The concept FSM to control the DMA Rx interface is given in Figure 46-38. DMA Rx FSM will move to Idle state if DMARXE[0] = 0.The TCD setting (typical case) is given in the next table. All other TCD fields equal zero. The minor loop transfers a single byte/ halfword as soon an entry is available in the Rx FIFO. A new software reset bit is

required that allows the LINFlexD FSMs to be reset in case this timeout state is reached or in any other case. The timeout counter can be re-written by software at any time to extend the timeout period.

Table 46-17. TCD setting – UART – Rx mode

TCD Field	Value		Description
	8 bits data	16 bits data	
CITER[14:0]	M		Multiple iterations for the "major" loop
BITER[14:0]	M		Multiple iterations for the "major" loop
NBYTES[31:0]	1	2	Minor loop transfer = 1 or 2 bytes
SADDR[31:0]	-		BDRM address
SOFF[15:0]	0		No increment (FIFO)
SSIZE[2:0]	0	1	Byte/Half-word transfer
SLAST[31:0]	0		
DADDR[31:0]	-		RAM address
DOFF[15:0]	1	2	Byte/Half-word increment
DSIZE[2:0]	0	1	Byte/Half-word transfer
DLAST_SGA[31:0]	–M	–M × 2	No scatter/gather processing
INT_MAJ	0/1		Interrupt disabled/enabled
D_REQ	1		Only on the last TCD of the chain.
START	0		No SW request

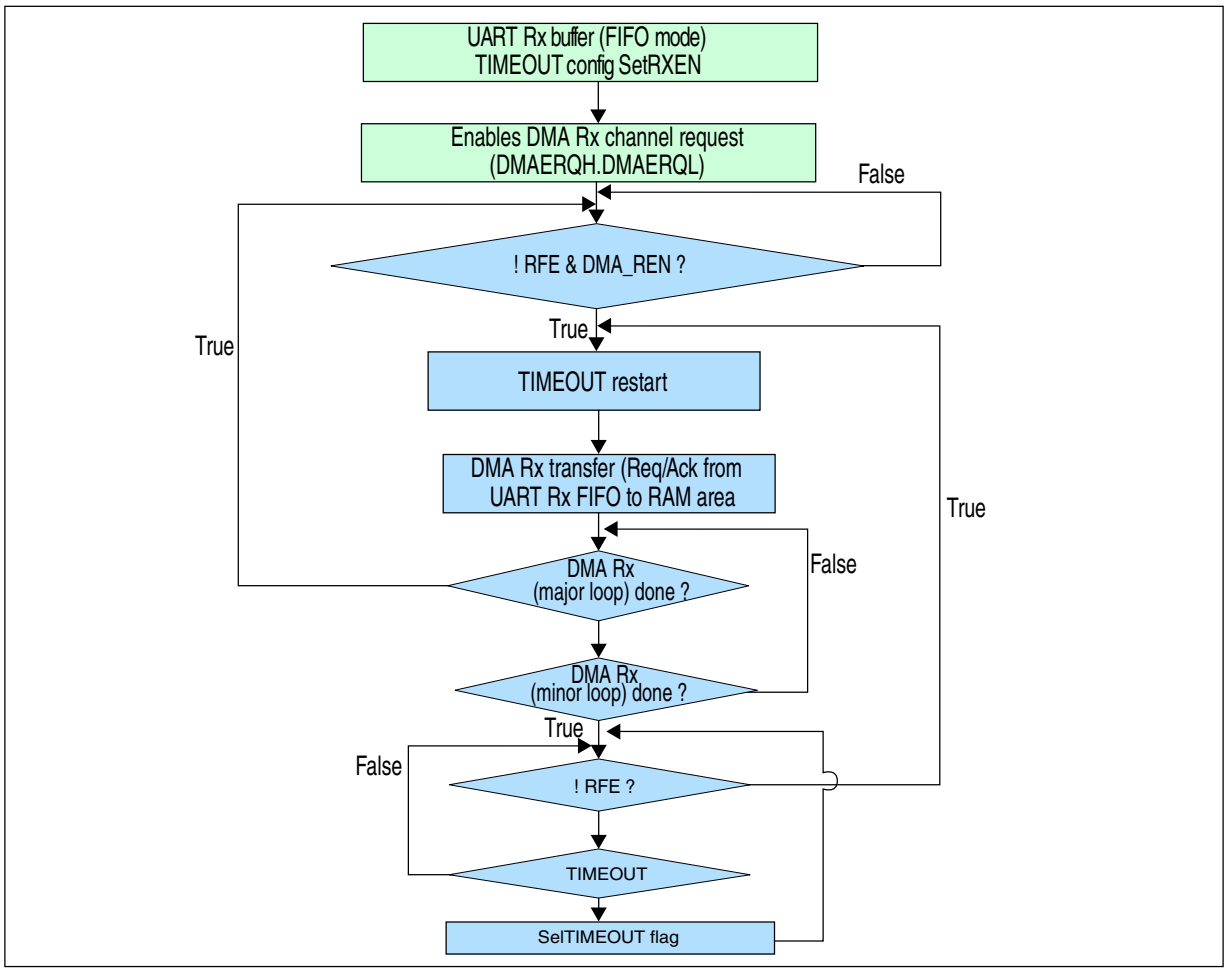


Figure 46-38. UART – DMA Rx FSM (concept scheme)

46.4.5.9 Use cases and limitations

- In LIN slave mode, the DMA capability can be used only if the ID filtering mode is activated. The number of ID filters enabled must be equal to the number of DMA channels enabled. The correspondence between channel number and ID filter is based on IFMI (identifier filter match index)
- In LIN master mode both the DMA channels (TX and RX) must be enabled in case the DMA capability is required
- In UART mode the DMA capability can be used only if the UART Tx/Rx buffers are configured as FIFOs
- DMA and CPU operating modes are mutually exclusive for the data/frame transfer on a UART or LIN node. Once a DMA transfer is finished the CPU can manage subsequent accesses

- Error management must always be executed via the CPU enabling the related error interrupt sources. DMA capability does not provide support for error management. Error management means checking status bits, handling IRQs, and potentially canceling DMA transfers
- The DMA programming model must be coherent with the TCD setting defined in this document
- When IPG_STOP is requested, SW has to first disable DMATXE/DMARXE channel registers, after the current minor loop finishes (indicated by the clearing of ACTIVE bit in DMA TCD register). This ensures that IPG_STOP_ACK is generated and IP enters STOP mode.

46.5 Memory map and register description

The following points should be considered for the below mentioned LINFlexD registers:

- Reset values are with-slave/without-slave (master only) format
- If not specified, then the bit can be configured for both master only and master/slave format; in other words, for both values of (generic) slave = 0 and (generic) slave = 1.

NOTE

In Master mode, the registers IFCR0-IFCR15 are not present and the corresponding absolute address of the below register changes as shown in the following table.

Table 46-18. Offsets of the register from offsets 8C to 9C

Register name	Master/slave mode	Master only mode
LINFlexD_GCR	0x8C	0x4C
LINFlexD_UARTPTO	0x90	0x50
LINFlexD_UARTCTO	0x94	0x54
LINFlexD_DMATXE	0x98	0x58
LINFlexD_DMARXE	0x9C	0x5C

NOTE

In Master Mode, read access to IFMI register and read/write access to IFER and IFMR registers would result in transfer error.

NOTE

Any access to A0 location will not generate a transfer error.

LINFlexD memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	LIN Control Register 1 (LINFlexD_LINCR1)	32	R/W	See section	46.5.1/1777
4	LIN Interrupt enable register (LINFlexD_LINIER)	32	R/W	0000_0000h	46.5.2/1780
8	LIN Status Register (LINFlexD_LINSR)	32	R/W	See section	46.5.3/1782
C	LIN Error Status Register (LINFlexD_LINESR)	32	w1c	0000_0000h	46.5.4/1786
10	UART Mode Control Register (LINFlexD_UARTCR)	32	R/W	0000_0000h	46.5.5/1787
14	UART Mode Status Register (LINFlexD_UARTSR)	32	R/W	0000_0000h	46.5.6/1793
18	LIN Time-Out Control Status Register (LINFlexD_LINTCSR)	32	R/W	0000_0200h	46.5.7/1795
1C	LIN Output Compare Register (LINFlexD_LINOCR)	32	R/W	0000_FFFFh	46.5.8/1797
20	LIN Time-Out Control Register (LINFlexD_LINTOCR)	32	R/W	See section	46.5.9/1798
24	LIN Fractional Baud Rate Register (LINFlexD_LINFBRR)	32	R/W	0000_0000h	46.5.10/ 1799
28	LIN Integer Baud Rate Register (LINFlexD_LINIBRR)	32	R/W	0000_0000h	46.5.11/ 1800
2C	LIN Checksum Field Register (LINFlexD_LINCFR)	32	R/W	0000_0000h	46.5.12/ 1800
30	LIN Control Register 2 (LINFlexD_LINCR2)	32	R/W	See section	46.5.13/ 1801
34	Buffer Identifier Register (LINFlexD_BIDR)	32	R/W	0000_0000h	46.5.14/ 1803
38	Buffer Data Register Least Significant (LINFlexD_BDRL)	32	R/W	0000_0000h	46.5.15/ 1804
3C	Buffer Data Register Most Significant (LINFlexD_BDRM)	32	R/W	0000_0000h	46.5.16/ 1805
40	Identifier Filter Enable Register (LINFlexD_IFER)	32	R/W	0000_0000h	46.5.17/ 1805
44	Identifier Filter Match Index (LINFlexD_IFMI)	32	R	0000_0000h	46.5.18/ 1806
48	Identifier Filter Mode Register (LINFlexD_IFMR)	32	R/W	0000_0000h	46.5.19/ 1806
4C	Identifier Filter Control Register (LINFlexD_IFCR0)	32	R/W	0000_0000h	46.5.20/ 1807
50	Identifier Filter Control Register (LINFlexD_IFCR1)	32	R/W	0000_0000h	46.5.20/ 1807
54	Identifier Filter Control Register (LINFlexD_IFCR2)	32	R/W	0000_0000h	46.5.20/ 1807
58	Identifier Filter Control Register (LINFlexD_IFCR3)	32	R/W	0000_0000h	46.5.20/ 1807
5C	Identifier Filter Control Register (LINFlexD_IFCR4)	32	R/W	0000_0000h	46.5.20/ 1807
60	Identifier Filter Control Register (LINFlexD_IFCR5)	32	R/W	0000_0000h	46.5.20/ 1807
64	Identifier Filter Control Register (LINFlexD_IFCR6)	32	R/W	0000_0000h	46.5.20/ 1807

Table continues on the next page...

LINFlexD memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
68	Identifier Filter Control Register (LINFlexD_IFCR7)	32	R/W	0000_0000h	46.5.20/ 1807
6C	Identifier Filter Control Register (LINFlexD_IFCR8)	32	R/W	0000_0000h	46.5.20/ 1807
70	Identifier Filter Control Register (LINFlexD_IFCR9)	32	R/W	0000_0000h	46.5.20/ 1807
74	Identifier Filter Control Register (LINFlexD_IFCR10)	32	R/W	0000_0000h	46.5.20/ 1807
78	Identifier Filter Control Register (LINFlexD_IFCR11)	32	R/W	0000_0000h	46.5.20/ 1807
7C	Identifier Filter Control Register (LINFlexD_IFCR12)	32	R/W	0000_0000h	46.5.20/ 1807
80	Identifier Filter Control Register (LINFlexD_IFCR13)	32	R/W	0000_0000h	46.5.20/ 1807
84	Identifier Filter Control Register (LINFlexD_IFCR14)	32	R/W	0000_0000h	46.5.20/ 1807
88	Identifier Filter Control Register (LINFlexD_IFCR15)	32	R/W	0000_0000h	46.5.20/ 1807
8C	Global Control Register (LINFlexD_GCR)	32	R/W	0000_0000h	46.5.21/ 1808
90	UART Preset Timeout Register (LINFlexD_UARTPTO)	32	R/W	0000_0FFFh	46.5.22/ 1810
94	UART Current Timeout Register (LINFlexD_UARTCTO)	32	R	0000_0000h	46.5.23/ 1811
98	DMA Tx Enable Register (LINFlexD_DMATXE)	32	R/W	0000_0000h	46.5.24/ 1812
9C	DMA Rx Enable Register (LINFlexD_DMARXE)	32	R/W	0000_0000h	46.5.25/ 1812

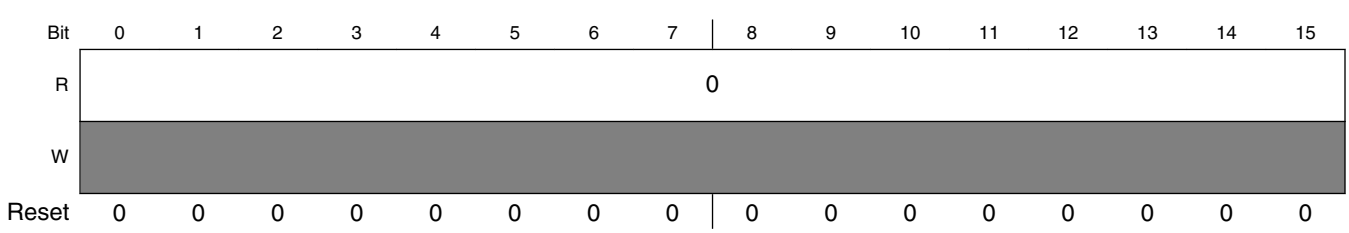
46.5.1 LIN Control Register 1 (LINFlexD_LINCR1)

LINCR1 consists of control bits used to configure features of the LINFlexD.

NOTE

When accessing the LINCR1 register, each reserved bit should be written to its original reset value.

Address: 0h base + 0h offset = 0h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CCD	CFD	LASE	AUTOWU	MBL				BF	Reserved	LBKM	MME	SSBL	RBLM	SLEEP	INIT
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	*	0	0	1	0

* Notes:
- MME field: If slave = 0, then MME bit is 1, else MME bit is 0

LINFlexD_LINCR1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CCD	Checksum Calculation disable This bit can be written during Initialization mode only. It is read-only in Normal mode. Register bit can be read in any mode, written only in initialization mode. 0 Checksum calculation is done by hardware. When this bit is reset the LINCFR register is read only. 1 Checksum calculation is disabled. When this bit is set the LINCFR register is read/write. User can program this register to sent a software calculated checksum/CRC (provided CFD is reset).
17 CFD	Checksum field disable This bit can be configured during Initialization mode only. It is read-only in Normal mode. Register bit can be read in any mode, written only in initialization mode. 0 Checksum field is sent after the required number of data bytes are sent 1 No checksum field is sent in the frame
18 LASE	LIN Autosynchronization Enable This bit can be programmed in Initialization mode only. It is read-only in Normal mode. (Note: If generic auto_sync = 0, then this bit will always read a 0). Register bit can be read in any mode, written only in initialization mode. 0 Autosynchronization disabled 1 Autosynchronization enabled
19 AUTOWU	Auto Wakeup This bit can be configured during Initialization mode only. This bit is utilized in UART mode also. Register bit can be read in any mode, written only in initialization mode. 0 Sleep bit is cleared by software only 1 Sleep bit gets cleared by hardware whenever WUF bit of LINSR is set
20–23 MBL	Master Break Length These bits choose the length of the Sync break to be generated by the master. These bits can be programmed in Initialization mode only. They are read-only in Normal mode. Register bit can be read in any mode, written only in initialization mode. 0000 10-bit break length

Table continues on the next page...

LINFlexD_LINCR1 field descriptions (continued)

Field	Description
	0001 11-bit break length 0010 12-bit break length 0011 13-bit break length 0100 14-bit break length 0101 15-bit break length 0110 16-bit break length 0111 17-bit break length 1000 18-bit break length 1001 19-bit break length 1010 20-bit break length 1011 21-bit break length 1100 22-bit break length 1101 23-bit break length 1110 36-bit (cooling) break length 1111 50-bit break length
24 BF	By-pass filter This bit can be programmed during Initialization mode only. **NOTE:** If generic slave = 0 or no_of_filters = 0, then this bit will always read a 1, and cannot be programmed. Register bit can be read in any mode, written only in initialization mode. 0 No IRQ if ID does not match any filter 1 A RX IRQ is generated on ID not matching any filter
25 Reserved	This field is reserved. When accessing the LINCR1 register, each reserved bit should be written to its original reset value.
26 LBKM	Loop Back mode Refer to "Loop back mode" in Test mode . Note that this bit can be programmed only in Initialization mode. This bit is utilized in UART mode also. Register bit can be read in any mode, written only in initialization mode. 0 Loop Back Mode disabled 1 Loop Back mode enabled
27 MME	Master mode enable This bit can be programmed in Initialization mode only. It is read-only in Normal mode. **NOTE:** If generic slave = 0, then this bit will read a '1' always, and cannot be programmed. Register bit can be read in any mode, written only in initialization mode. 0 Slave Mode 1 Master Mode
28 SSBL	Slave Mode Sync Break Length This bit can be programmed in Initialization mode only. It is read only in Normal mode. Register bit can be read in any mode, written only in initialization mode. 0 11 bit break length 1 10 bit break length

Table continues on the next page...

LINFlexD_LINCR1 field descriptions (continued)

Field	Description
29 RBLM	Receiver Buffer Locked mode This bit can be programmed in Initialization mode only. It is read-only in Normal mode. This bit is utilized in UART mode also. Register bit can be read in any mode, written only in initialization mode. 0 Receiver Buffer not locked, next incoming message will overwrite the old one 1 Receiver buffer locked against overrun. Once the buffer is full the next incoming message will be discarded if buffer is not released — in other words, RMB is not reset by software.
30 SLEEP	Sleep Mode Request This bit is set by software to request LINFlexD to enter Sleep mode. This bit is cleared by software or hardware (if AUTOWU bit in LINCR1 and WUF bit in LINSR are set) to exit sleep mode. This bit is utilized in UART mode also.
31 INIT	Initialization Mode Request The software sets this bit to switch the hardware into Initialization mode. On clearing this bit (and if SLEEP bit is also zero) LINFlexD enters normal mode. This bit is utilized in UART mode also.

46.5.2 LIN Interrupt enable register (LINFlexD_LINIER)

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZIE	OCIE	BEIE	CEIE	HEIE	0		FEIE	BOIE	LSIE	WUIE	Reserved	TOIE	DRIE	DTIE	HRIE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_LINIER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZIE	Stuck at zero Interrupt Enable An interrupt is generated if this bit is set and the Stuck at Zero Flag (SZF) in LINESR or UARTSR is set.

Table continues on the next page...

LINFlexD_LINIER field descriptions (continued)

Field	Description
	0 No interrupt 1 Interrupt enabled
17 OCIE	Output Compare Interrupt Enable 0 No interrupt 1 Interrupt generated when OCF bit in LINESR or UARTSR is set
18 BEIE	Bit Error Interrupt Enable 0 No interrupt 1 Interrupt generated when BEF bit in LINESR is set
19 CEIE	Checksum Error Interrupt Enable An interrupt is generated if this bit is set and the Checksum Error Flag (CEF) is set in LINESR. 0 No interrupt 1 Interrupt enabled
20 HEIE	Header Error Interrupt Enable An interrupt is generated when this bit is set and either of the following flags are set SFEF, SDEF, IDPEF in LINESR are set. **NOTE:** If generic slave = 0, then this bit will always read a 0 and cannot be programmed. 0 No interrupt 1 Interrupt enabled
21–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 FEIE	Frame Error Interrupt Enable 0 No interrupt 1 Interrupt generated if Frame Error Flag (FEF) bit is set in LINESR or UARTSR
24 BOIE	Buffer Overrun Error Interrupt Enable An interrupt is generated if this bit is set and the Buffer Overrun Flag (BOF) is set in LINESR or UARTSR. 0 No interrupt 1 Interrupt enabled
25 LSIE	LIN state Interrupt enable Interrupt is generated only when entering the above fields. This interrupt can mainly be used for debugging purposes. The interrupt has no status flag. 0 No interrupt 1 Interrupt generated on entering the following states: Sync Del, Sync Field, Identifier field, Checksum
26 WUIE	Wakeup interrupt enable If this bit is set and the WUF in LINSR or UARTSR is set then an interrupt is generated. 0 No interrupt 1 Interrupt enabled
27 Reserved	This field is reserved.

Table continues on the next page...

LINFlexD_LINIER field descriptions (continued)

Field	Description
28 TOIE	Timeout Interrupt Enable An interrupt is generated if this bit is set and UARTSR[TO] status bit is set (in UART mode). 0 No interrupt 1 Interrupt enabled
29 DRIE	Data Reception complete Interrupt enable An interrupt is generated when this bit is set and Data Received flag (DRF) in LINSR or UARTSR is set. 0 No interrupt 1 Interrupt enabled
30 DTIE	Data Transmitted Interrupt enable An interrupt is generated when this bit is set and Data Transmitted flag (DTF) in LINSR or UARTSR is set. 0 No interrupt 1 Interrupt enabled
31 HRIE	Header Received Interrupt An interrupt is generated when this bit is set and the Header Received flag (HRF) in LINSR is set. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed. 0 No interrupt 1 Interrupt enabled

46.5.3 LIN Status Register (LINFlexD_LINSR)

This register consists of status bits indicating the state of the LINFlexD hardware.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												AUTOSYNC_COMP	RDC		
W													w1c			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LINS				0		RMB	DRBNE	RXbusy	RDI	WUF	0		DRF	DTF	HRF
W	1						w1c	w1c			w1c			w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	*	0	0	0	0	0	0

* Notes:
- RDI field: Reset value of RDI reflects the RX pin state

LINFlexD_LINSR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 AUTOSYNC_COMP	AUTOSYNC_COMP This bit is set after autosynchronization is complete and when the LASE bit in LINCR1 register is enabled. Only after this bit is set, the contents of LINIBRR and LINFBRR registers can be read in autosynchronization mode. **NOTE:** If autosynchronization is disabled, this bit is reserved.
13–15 RDC	Receive Data Byte Count RDC contains the number of entries (bytes) in the Receive data buffer in LIN mode. RDCx is a read-only field available for debug purposes. 000 1 byte 001 2 bytes 010 3 bytes 011 4 bytes 100 5 bytes 101 6 bytes 110 7 bytes 111 8 bytes

Table continues on the next page...

LINFlexD_LINSR field descriptions (continued)

Field	Description
16–19 LINS	LIN state In UART mode Idle, Init, Sleep, and Data Transmission/Reception states are flagged by the LIN status bits. **NOTE:** The value of this bit field doesn't change by any write operation to it. Writing 0xF to this bit field clears the Rx interrupt, only when set due to the LIN state event. 0000 Sleep mode: LINFlexD is in Sleep mode, to save power consumption. 0001 Init mode: LINFlexD is in Initialization mode. 0010 Idle mode: This mode is entered when: SLEEP bit and INIT bit are reset by software; Wakeup pulse has been received on RX pin (AUTOWU set); Previous frame transmission/reception has been completed/aborted 0011 Sync break: In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In slave mode, a falling edge followed by a dominant state has been detected. Receiving sync break. In master mode, sync break transmission ongoing. Note: In slave mode upon any error LIN state could be either Idle or Rec Break, depending on last bit detected on LIN_RX. If last bit detected is dominant then Rec_Break, otherwise Idle. 0100 Sync Del: In Slave mode, valid Sync break has been detected (10 bit or 11 bit). Waiting for a rising edge. In Master mode, Sync break transmission has been completed, sync delimiter transmission is ongoing. 0101 Sync Field: In Slave mode, a valid sync Del has been detected (recessive state for at least one bit time). Receiving sync field. In Master mode, sync field transmission ongoing. 0110 Identifier Field: In Slave mode, a valid sync field has been received. Receiving ID field. In Master mode, identifier transmission is ongoing. 0111 Header Reception/Transmission: In Slave mode, a valid header has been received and Identifier field is available in the BIDR. In Master mode, header transmitted. 1000 Data Reception/Data Transmission: In Receiver mode, reception ongoing. In Transmitter mode, response transmission ongoing. 1001 Checksum: Data transmission/reception completed, checksum transmission/reception ongoing.
20–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 RMB	Release Message Buffer 0 Buffer data is free. Reset by hardware in when in Initialization mode 1 Buffer data ready to be read by software. This bit should be cleared by software after reading the data received in the buffer.
23 DRBNE	Data Reception Buffer Not Empty Flag This bit is set by hardware as soon as the first byte of response has been received and stored in BDRL (when there is at least one data byte in reception buffer). Software should clear it after reading all the buffers. This bit is also reset by hardware in Initialization mode. This flag could be checked by software in case of a response timeout event.
24 RXbusy	Receiver Busy flag In Slave mode after header reception, if DIR bit is reset and reception starts, then this bit is set. In this case user cannot set the DTRQ bit. 0 Receiver is idle 1 Reception ongoing

Table continues on the next page...

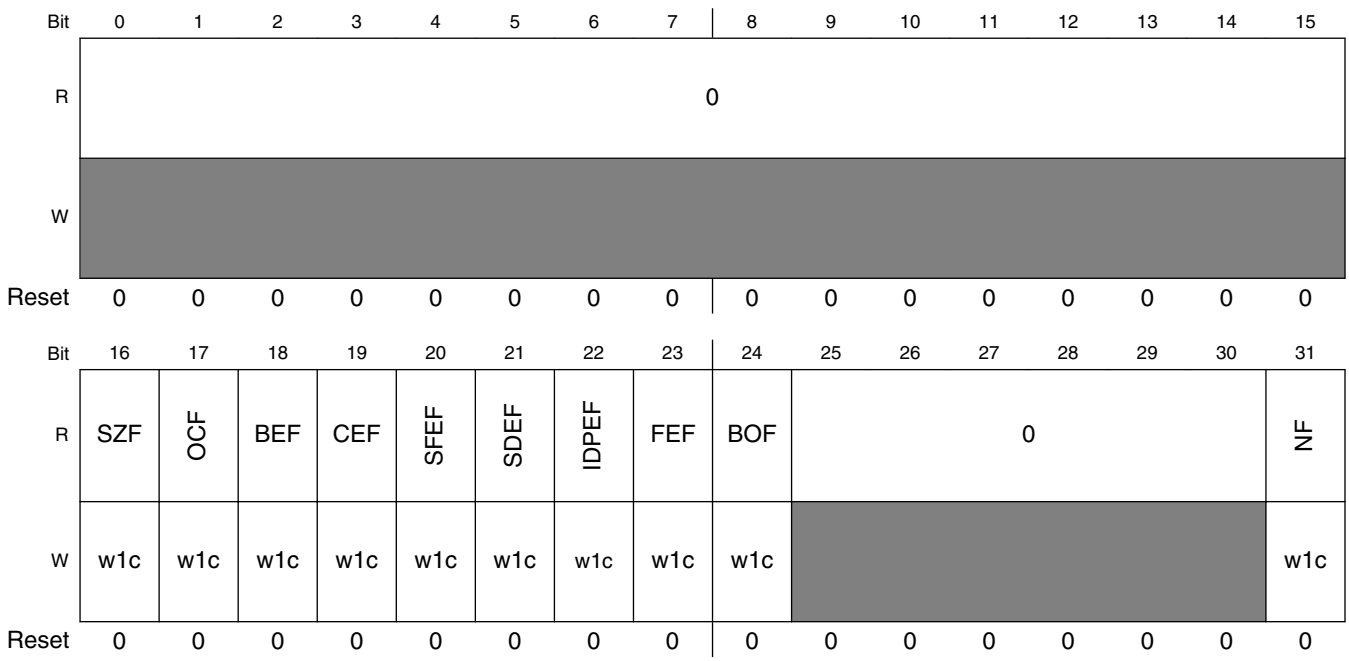
LINFlexD_LINSR field descriptions (continued)

Field	Description
25 RDI	LIN Receive signal This bit reflects the current status of the Rx pin. **NOTE:** After reset is released, RDI reflects the actual value of Rx pin.
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the Rx pin. 1. When slave is in Sleep mode 2. When master is in Sleep mode or Idle mode It can be cleared by software. It gets reset by hardware in Initialization mode.
27–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 DRF	Data Reception Completed flag This bit is set by hardware and indicates that data reception has been completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **NOTE:** In case framing error or checksum error occurs then this flag is not set.
30 DTF	Data Transmission Completed flag This bit is set by hardware and indicates that data transmission is completed. This flag should be cleared by software. This bit is also reset by hardware in Initialization mode. **NOTE:** For LIN mode, in case a bit error occurs (and IOBE is 1) then this flag is not set.
31 HRF	Header Received flag This bit is set when the header reception is completed. It should be cleared by software. This bit is set only when: a) All filters are inactive and Bypass filter (BF) bit is set b) No match in any filter and Bypass filter (BF) bit is set c) TX filter match At end of frame or frame aborted, if HRF is still set, it gets reset by hardware. This bit is also reset by hardware in Initialization mode. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.

46.5.4 LIN Error Status Register (LINFlexD_LINESR)

Refer to Errors for detailed description of all errors.

Address: 0h base + Ch offset = Ch



LINFlexD_LINESR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZF	Stuck at Zero flag This bit is set when there is a stuck-at-zero timeout error. It should be reset by software.
17 OCF	Output Compare Flag 0: No output compare event occurred 1: In master mode, LINESR[OCF] flag is set when counter LINTCSR[CNT] has matched the content of LINOCR[OC2]. In slave mode, LINESR[OCF] is set when the content of the counter LINTCSR[CNT] has matched the content of LINOCR[OC1] or LINOCR[OC2]. This bit should be cleared by software. If this bit is set, MODE bit in LINTCSR is cleared, and the IOT bit of LINTCSR is set, then LINFlexD moves to Idle state. If the MODE bit in LINTCSR is clear, then OCF gets reset by hardware in Initialization mode; if the MODE bit is set, then OCF maintains its status irrespective of the LIN state.
18 BEF	Bit Error flag This bit is set by hardware when there is a bit error. It should be cleared by software. Reset by hardware in Initialization mode.

Table continues on the next page...

LINFlexD_LINESR field descriptions (continued)

Field	Description
19 CEF	Checksum Error flag This bit is set by hardware if the received checksum does not match the hardware-calculated checksum. It should be cleared by software. This error will never occur if CCD or CFD bit of LINCR1 is set. Reset by hardware in Initialization mode.
20 SFEF	Sync Field Error flag This bit is set by hardware when the received Sync Field is inconsistent. It should be cleared by software. Reset by hardware in Initialization mode. If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
21 SDEF	Sync Delimiter Error flag This bit is set by hardware when the delimiter is too short (in other words, less than one bit time). It should be cleared by software. Reset by hardware in Initialization mode. **NOTE:** If generic slave = 0, then this bit will always read a 0, and cannot be programmed.
22 IDPEF	ID Parity Error flag This bit is set by hardware when there is an error in the ID parity. It should be cleared by software. **NOTE:** Header Error Interrupt is triggered for SFEF or SDEF or IDPEF flag is set and HEIE is set. If generic slave = 0, then this bit will always read a 0, and cannot be programmed. For generic slave = 1,this bit is reset by hardware in Initialization mode.
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. Reset by hardware in initialization mode.
24 BOF	Buffer overrun flag This bit is set by hardware when there is a new byte received and RMB bit is not cleared. It can be cleared by software. Reset by hardware in initialization mode.
25–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. Reset by hardware in Initialization mode.

46.5.5 UART Mode Control Register (LINFlexD_UARTCR)

NOTE

When accessing the UARTCR register, reserved bits should always be written to zero.

NOTE

- The LINFlexD module in UART mode does not support communication with Special Word Length (UARTCR[WLS] = 1) in buffer mode.
- The LINFlexD module in UART mode supports communication with Special Word Length (UARTCR[WLS] = 1) in FIFO mode only with UARTCR[WL1, WL0] = 1,1.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MIS	CSP			OSR				ROSE	NEF			DTU_PCETX	SBUR		WLS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TDFL_TFC			RDFL_RFC			RFBM	TFBM	WL1	PC1	RxEn	TxEn	PC0	PCE	WL0	UART
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_UARTCR field descriptions

Field	Description
0 MIS	Monitor Idle State Register bit can be read in any mode, written only in initialization mode. 0 UARTCTO monitors the number of bits to be received. 1 UARTCTO monitors the idle state of the reception line.
1–3 CSP	Configurable Sample Point (i) These bits will decide the sample point during reduced over sampling. CSP can take the following range of values for a certain over sampling rate: **OSR** / **CSP** 4 / 2,3 5 / 2, 3, 4 6 / 3, 4, 5 8 / NA Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description
4–7 OSR	Over Sampling Rate These bits are programmable by the user to configure the number of samples taken for a bit when reduced over sampling is enabled. Allowed values are: 4, 5, 6 and 84 and 8 . Register bit can be read in any mode, written only in initialization mode.
8 ROSE	Reduced Over Sampling EnableRegister bit can be read in any mode, written only in initialization mode. 0 Each bit is over sampled sixteen times. 1 OSR bits decide the over sampling rate.
9–11 NEF	Number of expected frame These bits are used to configure the number of expected frames in UART reception mode. If the DTU bit is set, then the UART timeout counter will be reset after the configured number of frames have been received. Register bit can be read in any mode, written only in Initialization mode. Register bit is a read/write field.
12 DTU_PCETX	Disable Timeout in UART mode This bit can be programmed in Initialization mode only when the UART bit is set. There is an internal counter that gives the number of frames received. **In Buffer mode:** This timer gets reset whenever the number of bytes received (rec_byte_cnt) is equal to RDFL. At this point the DRF bit is also set and num_of_frames (number of frames received), rec_byte_cnt are reset. When num_of_frames equals NEF, Disable Timeout is generated. But this clears num_of_frames also. Thus the timer restarts again. The value of counter is 0 when it restarts. **In FIFO mode:** When num_of_frames equals NEF, Disable Timeout is generated which clears num_of_frames and causes the timer to restart again. The value of counter is 0 when it restarts. Notes: • Disable Timeout causes only the Timer to restart which enables Timeout again. • Timer reset means it resets and starts counting again and the Timeout is enabled. 0 Timeout has to be handled by software 1 Timeout in UART mode is disabled after the configured number of data frames are received
13–14 SBUR	Stop bits in UART reception mode When the UART is used for transmission and reception we have to set the same number of stop bits in GCR and SBUR. When the UART is used only as receiver, it is enough to set SBUR bits only. This bit can be programmed in the initialization mode only, when the UART bit is set. Register bit can be read in any mode, written only in Initialization mode. 00 1 stop bit 01 2 stop bits 10 3 stop bits 11 reserved
15 WLS	Special Word Length in UART mode

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description
	This bit can be programmed in initialization mode only when the UART bit is set. If this bit is set, setting WL and PCE bits has no effect in UART reception although parity check is enabled by default in this mode and PC0/1 bits can be used to select the parity control. This bit is given the highest priority in UART reception mode. Register bit can be read in any mode, written only in Initialization mode. 0 This bit is disabled. 1 This bit enables 12-bit + parity bit in reception (UART mode) for MSC (Micro Second Channel upstream frame) support.
16–18 TDFL_TFC	Transmitter Data Field Length/TX FIFO Counter TDFL defines the number of bytes to be transmitted in UART buffer mode (TFBM = 0). TDFL is a read/write field. Bit 16 is reserved and not implemented. When the UART data length is configured as halfword (WL = 10 or WL = 11), only the configurations TDFL = x01 or TDFL = x11 are allowed. x00 : 1 byte x01 : 2 bytes x10 : 3 bytes x11 : 4 bytes Register bit is a read/write field. TFC contains the number of entries (bytes) of the Tx FIFO in UART FIFO mode (TFBM = 1). TFCx is a read-only field available for debug purposes. 000 : empty 001 : 1 byte 010 : 2 bytes 011 : 3 bytes 100 : 4 bytes Others : reserved TDFLTFC can be programmed and are significative only when the UART bit is set. Register bit can be read in any mode, written only in Initialization mode. **NOTE:** When a debugger is connected, this counter will decrement if TxFIFO is being read through the debugger. **NOTE:** In case of data path is functioning normally, RFC/TFC counters will be cleared internally by HW.
19–21 RDFL_RFC	Reception Data Field Length /RX FIFO Counter RDFL defines the number of bytes to be received in UART buffer mode (RFBM = 0). RDFL is a read/write field. Bit 19 is reserved and not implemented. When the UART data length is configured as halfword (WL = 10 or WL = 11), only the configurations RDFL = x01 or RDFL = x11 are allowed. x00 : 1 byte x01 : 2 bytes x10 : 3 bytes x11 : 4 bytes RFC contains the number of entries (bytes) of the Rx FIFO in UART FIFO mode (RFBM = 1). RFCx is a read-only field available for debug purposes. 000 : Empty

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>001 : 1 byte
010 : 2 bytes/1.5 byte in case of WLS bit setting
011 : 3 bytes
100 : 4 bytes/2x1.5 byte in case of WLS bit setting
Others - Reserved
RDFLRFC can be programmed and are significative only when the UART bit is set.
NOTE: In buffer mode, RDFL should be programmed to be greater than or equal to NEF (number of expected frames). In FIFO mode, there is no such constraint.
NOTE: When a debugger is connected, this counter will decrement if RxFIFO is being read through the debugger.
NOTE: In case of data path is functioning normally, RFC/TFC counters will be cleared internally by HW.</td></tr>
<tr><td>22
RFBM</td><td>RFBM Rx Fifo/Buffer mode
This bit can be programmed in Initialization mode, only when the UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
0 Rx Buffer mode enabled
1 Rx Fifo mode enabled (mandatory in DMA Rx mode)</td></tr>
<tr><td>23
TFBM</td><td>Tx Fifo/Buffer mode
This bit can be programmed in initialization mode, only when the UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
0 Tx Buffer mode enabled
1 Tx Fifo mode enabled (mandatory in DMA Tx mode)</td></tr>
<tr><td>24
WL1</td><td>Word Length in UART mode
This bit can be programmed in Initialization mode only, when the UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table>
<tr><th>WL1</th><th>WL0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>7 bits data + parity</td></tr>
<tr><td>0</td><td>1</td><td>8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1</td></tr>
<tr><td>1</td><td>0</td><td>15 bits data + parity</td></tr>
<tr><td>1</td><td>1</td><td>16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1</td></tr>
</table></td></tr>
<tr><td>25
PC1</td><td>Parity Control
This bit can be programmed in Initialization mode, only when UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table>
<tr><th>PC1</th><th>PC0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>Parity sent is Even</td></tr>
</table></td></tr>
</table>

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>PC1</th><th>PC0</th><th>Description</th></tr>
<tr><td>0</td><td>1</td><td>Parity sent is Odd</td></tr>
<tr><td>1</td><td>0</td><td>A logical 0 is always transmitted/ checked as parity bit</td></tr>
<tr><td>1</td><td>1</td><td>A logical 1 is always transmitted/ checked as parity bit</td></tr>
</table>
</td></tr>
<tr><td>26
RxEn</td><td>Receiver Enable
This bit can be programmed only when the UART bit is set.
0 Receiver disabled
1 Receiver enabled</td></tr>
<tr><td>27
TxEn</td><td>Transmitter Enable
This bit can be programmed only when UART bit is set.
0 Transmitter disabled
1 Transmitter enabled, transmission starts only when this bit is set and Data byte 0 (DATA0) is programmed</td></tr>
<tr><td>28
PC0</td><td>Parity Control
This bit can be programmed in Initialization mode, only when UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table>
<tr><th>PC1</th><th>PC0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>Parity sent is Even</td></tr>
<tr><td>0</td><td>1</td><td>Parity sent is Odd</td></tr>
<tr><td>1</td><td>0</td><td>A logical 0 is always transmitted/ checked as parity bit</td></tr>
<tr><td>1</td><td>1</td><td>A logical 1 is always transmitted/ checked as parity bit</td></tr>
</table>
</td></tr>
<tr><td>29
PCE</td><td>Parity Control Enable
This bit can be programmed in Initialization mode only, when the UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
0 Parity transmit/check Disable
1 Parity transmit/check Enable</td></tr>
<tr><td>30
WL0</td><td>Word Length in UART mode
This bit can be programmed in Initialization mode only, when UART bit is set.
Register bit can be read in any mode, written only in initialization mode.
<table>
<tr><th>WL1</th><th>WL0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>7 bits data + parity</td></tr>
</table>
</td></tr>
</table>

Table continues on the next page...

LINFlexD_UARTCR field descriptions (continued)

Field	Description		
	WL1	**WL0**	**Description**
	0	1	8 bits data when PCE = 0 or 8 bits data + parity when PCE = 1
	1	0	15 bits data + parity
	1	1	16 bits data when PCE = 0 or 16 bits data + parity when PCE = 1
31 UART	UART Mode This bit can be programmed in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. 0 LIN mode 1 UART mode		

46.5.6 UART Mode Status Register (LINFlexD_UARTSR)

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SZF	OCF	PE				RMB	FEF	BOF	RDI	WUF	RFNE	TO	DRFRFE	DTFTFF	NF
W	w1c	w1c	w1c				w1c	w1c	w1c	w1c	w1c		w1c			w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_UARTSR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SZF	Stuck at Zero flag This bit is set by hardware when 100 dominant bits are detected. It should be cleared by software. An interrupt will be generated if the SZIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
17 OCF	Output Compare Flag An interrupt will be generated if the OCIE bit in LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set. **NOTE:** For Baud rate above 1 Mbit/s, this flag is not usable. 0 No output compare event occurred 1 The content of the counter has matched the content of LINOCR
18–21 PE	Parity Error flag These bits indicate if there is a Parity Error in the corresponding byte. No interrupt is generated if this error occurs. This bit will reflect the same value as in LINESR, when in initialization mode and UART bit set. **NOTE:** Parity is checked only after complete frame is received. Following are the conditions when WL bits = 01 or 11 (either PE0 or PE2 is only set). • When PE0 is set, it indicates Parity error in either first or second byte received • When PE2 is set, it indicates Parity error in either third or fourth byte received 0 No parity error 1 Parity error in the corresponding received byte
22 RMB	Release Message Buffer This bit should be cleared by software. This bit will reflect the same value as in LINSR, when in Initialization mode and the UART bit is set. 0 Buffer data is free 1 Buffer data ready to be read by software
23 FEF	Framing Error flag This bit is set by hardware when there is a framing error (invalid stop bit). It should be cleared by software. It will generate an interrupt if the FEIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
24 BOF	FIFO/Buffer overrun flag This bit is set by hardware when there is a new byte received and the RMB bit is not cleared in UART buffer mode. In UART FIFO mode this bit is set when there is a new byte and the Rx FIFO is full. In UART FIFO mode, once Rx FIFO is full, the new received message will be discarded irrespective of the new value of the RBLM bit. In UART Rx Buffer mode, if RBLM is set then the new message received will be discarded; if RBLM is reset then the new message will overwrite the buffer. It can be cleared by software writing a 1. An interrupt is generated if the BOIE bit of LINIER is set. This bit will reflect the same value as in LINESR, when in Initialization mode and the UART bit is set.
25 RDI	Receiver Data Input signal This bit reflects the current status of the RX pin.

Table continues on the next page...

LINFlexD_UARTSR field descriptions (continued)

Field	Description
26 WUF	Wakeup flag This bit is set by hardware when a falling edge is detected on the RX pin in sleep mode. It should be cleared by software. An interrupt will be generated if the WUIE bit in LINIER is set. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set.
27 RFNE	Receive FIFO Not Empty RFNE bit is set by hardware in UART FIFO mode (RFBM = 1), when there is at least one data byte present in the receive FIFO. RFNE is a read-only bit for debugging purposes. This flag can be used by software in case of a timeout event.
28 TO	Timeout This bit is set by hardware when a UART timeout occurs — in other words, the value of UARTCTO becomes equal to the preset value of the timeout (UARTPTO register setting). TO should be cleared by software. The SR bit should be used to reset the receiver fsm to Idle state in case of UART TIMEOUT for UART reception, depending on the application, in both buffer and FIFO mode. An interrupt will be generated when LINIER.TOIE bit is set on the Error interrupt line in UART mode.
29 DRFRFE	Data Reception Completed Flag /Rx FIFO Empty Flag DRF is set by hardware in UART buffer mode (RFBM = 0) and indicates that the number of bytes programmed in RDFL have been received. DRF should be cleared by software. An interrupt will be generated if the DRIE bit in LINIER is set. DRF bit is set when the configured number of valid stop bits are received for the last frame (number of frames is configurable by RDFL bits). DRF is set irrespective of framing error in case framing error is in the last STOP bit configured, parity error or overrun error. This bit will reflect the same value as in LINSR when in Initialization mode and the UART bit is set. Register bit can be read/cleared by software. Writing 1 clears contents. RFE is set by hardware in UART FIFO mode (RFBM = 1) when the RX FIFO is empty. RFE is a read-only bit for debugging purposes. It is internally used by the DMA RX interface.
30 DTFTFF	Data Transmission Completed Flag/ TX FIFO Full Flag DTF is set by hardware in UART buffer mode (TFBM = 0) and indicates that data transmission is completed. DTF should be cleared by software. An interrupt will be generated if the DTIE bit in LINIER is set. This bit will reflect the same value as in LINSR when in Initialization mode and the UART bit is set. Register bit can be read/cleared by software. Writing 1 clears contents. TFF is set by hardware in UART FIFO mode (TFBM = 1) when TX FIFO is full. TFF is a read-only bit for debugging purposes. It is internally used by the DMA TX interface.
31 NF	Noise flag This bit is set by hardware when noise is detected in the received character. It should be cleared by software. This bit will reflect the same value as in LINESR when in Initialization mode and the UART bit is set. During reduced oversampling (ROSE bit = 1), it is enabled only when OSR = 8.

46.5.7 LIN Time-Out Control Status Register (LINFlexD_LINTCSR)

This register contains control and status bits for timeout feature.

NOTE

When the MODE bit is 0, any activity on the transmit or receive pins will cause an unwanted change in the value of the 8-bit field Output Compare Value 2 (OC2) of the LIN Output Compare register (LINOCR). The LINFlexD module is enabled in LIN mode and the value of the MODE bit of the LIN Timeout Control Status register (LINTCSR) is changed from ‘1’ to ‘0’, then the old value of the Output Compare Value 1 (OC1) and Output Compare Value 2 (OC2) of the LIN Output Compare register (LINOCR) is retained. As a consequence, if the module is reconfigured from UART to LIN mode, or LINTCSR MODE bit is changed from ‘1’ to ‘0’, an incorrect timeout exception is generated when a LIN communication starts. To avoid this, set mode bit to '1', then reconfigure the LINFlexD in LIN mode and then reset the MODE bit. Before switching LINTCSR[MODE] from 1 to 0 in between frames, load LINOCR with 0xFFFF.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					MODE	IOT	TOCE	CNT							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

LINFlexD_LINTCSR field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 MODE	Time-out counter mode This bit can be configured only during initialization. Register bit can be read in any mode, written only in initialization mode. **NOTE:** Always set LINTCSR[MODE] to 1 when UARTCR[UART] is 1. **NOTE:** Before switching LINTCSR[MODE] from 1 to 0 in between frames, load LINOCR to 0xFFFF. 0 LIN mode 1 Output compare mode
22 IOT	Idle on timeout Register bit can be read in any mode, written only in initialization mode. This feature is applicable only when MODE bit in LINTCSR is cleared. 0 LIN state machine does not reset to Idle on timeout 1 LIN state machine resets to Idle on timeout event
23 TOCE	Time-out counter enable TOCE is always configurable by software in Initialization mode. If LIN state is other than INIT and if timer is configured in LIN mode, then hardware takes control of TOCE. 0 Time-out counter disable. OCF flag is not set on an output compare event. 1 Time-out counter enable. OCF flag is set if an output compare event occurs.
24–31 CNT	Counter Value These bits reflect the value of a counter used for timeout. **NOTE:** For proper functionality of this counter, LINIBRR should be >= 5.

46.5.8 LIN Output Compare Register (LINFlexD_LINOCR)

This register contains the value to be compared to the LINTCSR:CNT value. This register is writable by software only in Output Compare Mode.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																OC2								OC1							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

LINFlexD_LINOCR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–23 OC2	Output compare value 2
24–31 OC1	Output compare value 1

46.5.9 LIN Time-Out Control Register (LINFlexD_LINTOCR)

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				RTO				0	HTO						
W																
Reset	0	0	0	0	1	1	1	0	0	*	*	*	*	*	*	*

* Notes:

- HTO field: HTO reset values:

  HTO resets to 0011100b in Master only mode, if generic slave = 0.

  HTO resets to 0101100b in Master/Slave mode, if generic slave = 1.

LINFlexD_LINTOCR field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 RTO	Response timeout value This is the response timeout duration (in bit time) for 1 byte. The reset value is 0Eh = 14, corresponding to $T_{Response_Maximum} = 1.4 \times T_{Response_Nominal}$
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 HTO	Header timeout value This register contains the header timeout duration (in bit time). This register can be written only for Slave mode. If Master mode is enabled then these bits will always reflect 28 (1.4 x 10 bits of sync + 1.4 x 10 bits of ID). For slave, these bits should be programmed without considering 11 bits of break.

46.5.10 LIN Fractional Baud Rate Register (LINFlexD_LINFBRR)

This register consists of bits that decide the fractional part of the LIN Baud Rate. It can be programmed only in Initialization mode.

NOTE

When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

NOTE

This register cannot be used when reduced oversampling is enabled (ROSE bit = 1)

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												FBR			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_LINFBRR field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 FBR	Fractional Baud rates Register bit can be read in any mode, written only in initialization mode. 0000 Fraction(LDIV) = 0 0001 Fraction(LDIV) = 1/16 0010 Fraction(LDIV) = 2/16 0011 Fraction(LDIV) = 3/16 0100 Fraction(LDIV) = 4/16 0101 Fraction(LDIV) = 5/16 0110 Fraction(LDIV) = 6/16 0111 Fraction(LDIV) = 7/16 1000 Fraction(LDIV) = 8/16 1001 Fraction(LDIV) = 9/16 1010 Fraction(LDIV) = 10/16 1011 Fraction(LDIV) = 11/16 1100 Fraction(LDIV) = 12/16 1101 Fraction(LDIV) = 13/16 1110 Fraction(LDIV) = 14/16 1111 Fraction(LDIV) = 15/16

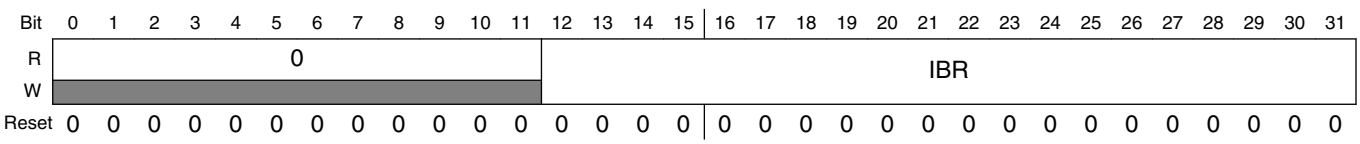
46.5.11 LIN Integer Baud Rate Register (LINFlexD_LINIBRR)

This register consists of control bits that decide the baud rate along with the LINFBRR. It can be programmed only in Initialization mode.

NOTE

When LASE bit is set, this register should be read only after AUTOSYNC_COMP bit in LINSR register is set to obtain the correct value.

Address: 0h base + 28h offset = 28h



LINFlexD_LINIBRR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–31 IBR	Integer Baud rates These bits along with the fractional baud rate bits decide the LIN baud rate. IBR = 0h: LIN clock disabled IBR = 1h: Mantissa (LDIV) = 1 ... IBR = FFFFEh: Mantissa (LDIV) = 1048574 IBR = FFFFFh: Mantissa (LDIV) = 1048575 Register bit can be read in any mode, written only in initialization mode.

46.5.12 LIN Checksum Field Register (LINFlexD_LINCFR)

NOTE

There is a delay between 4 to 6 clock cycles of PBRIDGEx_CLK for the internal checksum's value (which is clocked with LIN_CLK/16 * LDIV) to reflect on LINCFR.

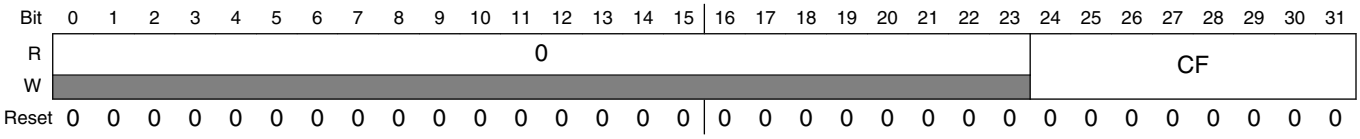
This register consists of checksum bits.

CFD	CCD	LINCHKSUM Read/write	Checksum sent
1	1	read/write	None

Table continues on the next page...

CFD	CCD	LINCHKSUM Read/write	Checksum sent
1	0	read-only	None
0	1	read/write	Programmed checksum
0	0	read-only	Calculated checksum

Address: 0h base + 2Ch offset = 2Ch



LINFlexD_LINCFR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 CF	Checksum bits When the CCD bit is reset these bits are read-only and are calculated by hardware. When the CCD bit is set, these bits can be written by software.

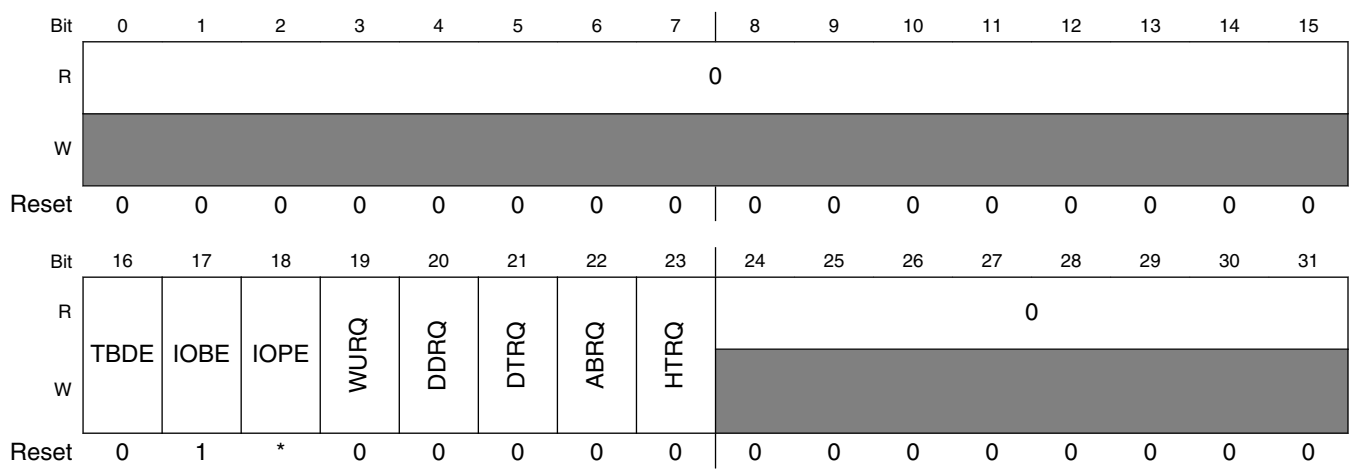
46.5.13 LIN Control Register 2 (LINFlexD_LINCR2)

This register includes control status bits related to buffer operations.

NOTE

When accessing the LINCR2 register, each reserved bit should be written to its original reset value.

Address: 0h base + 30h offset = 30h



* Notes:

- IOPE field: When slave = 0, this field always reads ‘0’ and cannot be programmed else this bit is programmable and reset value is 1.

LINFlexD_LINCR2 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 TBDE	Two Bit delimiter bit This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. 0 Delimiter length in break field is 1 bit 1 Delimiter length in break field is 2 bits
17 IOBE	Idle on Bit Error This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. 0 Bit Error does not reset LIN state machine 1 Bit Error resets LIN state machine
18 IOPE	Idle on Identifier Parity Error This bit can be set in Initialization mode only. Register bit can be read in any mode, written only in initialization mode. 0 Parity Error does not reset LIN state machine 1 Parity Error resets LIN state machine
19 WURQ	Wakeup Generate Request Setting this bit will generate a wakeup pulse. It is reset by hardware when the wakeup character has been transmitted. The character sent during wakeup is copied from BDRL (DATA0). Note that this bit cannot be set in Sleep mode — software has to exit Sleep mode before setting this bit. Bit Error is not checked when transmitting the wakeup request. Register bit can be read/set by software
20 DDRQ	Data Discard request Set by software to stop data reception if the frame does not concern the node. This bit is reset by hardware once LINFlexD ignores the response and moves to Idle state. For LIN slave this bit can be set only when HRF bit is set and Identifier is software-filtered. Register bit can be read/set by software
21 DTRQ	Data Transmission Request Set by software in slave mode to request the transmission of the LIN Data field stored in the Buffer data register. This bit can be set only when the HRF bit is set (to ensure that data transmission is requested only after a header reception). Cleared by hardware when the request has been completed, or on abort request or error condition. In Master mode, this bit is set by hardware when the DIR bit is set and header transmission is complete. Register bit can be read/set by software
22 ABRQ	Abort Request Set by software to abort the current transmission.

Table continues on the next page...

LINFlexD_LINCR2 field descriptions (continued)

Field	Description
	Cleared by hardware when the transmission has been aborted. LINFlexD aborts the transmission at the end of the current bit. This bit can abort a wakeup request also and can be used in UART mode also. Register bit can be read/set by software
23 HTRQ	Header Transmission Request Set by software to request the transmission of the LIN Header. Cleared by hardware when the request has been completed or on abort request. This bit has no effect in UART mode. **NOTE:** In master mode, if both HTRQ and ABRQ are set at the same time then ABRQ has no effect. Similarly, in slave mode after header reception, if DTRQ and ABRQ are simultaneously set then ABRQ has no effect.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

46.5.14 Buffer Identifier Register (LINFlexD_BIDR)

This register contains the bits which provide information about the identifier of the transaction and other related information.

NOTE

All the fields (ID, CSS, DIR, DFL) of the BIDR register must be updated when an ID filter (enabled) in Slave mode (Tx or Rx) matches the ID received.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved			DFL			DIR	CCS	0		ID					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_BIDR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–18 Reserved	This field is reserved.
19–21 DFL	Data Field Length Number of data bytes in the response part of the frame.

Table continues on the next page...

LINFlexD_BIDR field descriptions (continued)

Field	Description
	DFL = Number of data bytes - 1
22 DIR	Direction This bit controls the direction of the data field. 0 LINFlexD receives the data and copy them in the BDR registers 1 LINFlexD transmits the data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data filed only. This is compatible with LIN specification 1.3 and lower.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 ID	Identifier Identifier part of the identifier field without the identifier parity. This field can be written only in Master mode (MME = '1').

46.5.15 Buffer Data Register Least Significant (LINFlexD_BDRL)

This register is a part of an 8-byte data buffer.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DATA3								DATA2								DATA1								DATA0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

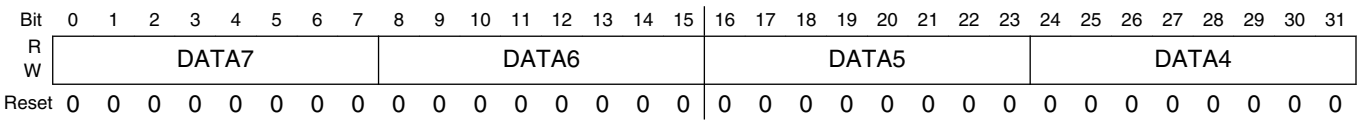
LINFlexD_BDRL field descriptions

Field	Description
0–7 DATA3	Data Byte 3 Data byte 3 of the data field.
8–15 DATA2	Data Byte 2 Data byte 2 of the data field.
16–23 DATA1	Data Byte 1 Data byte 1of the data field.
24–31 DATA0	Data Byte 0 Data byte 0 of the data field.

46.5.16 Buffer Data Register Most Significant (LINFlexD_BDRM)

This register is a part of an 8-byte data buffer.

Address: 0h base + 3Ch offset = 3Ch



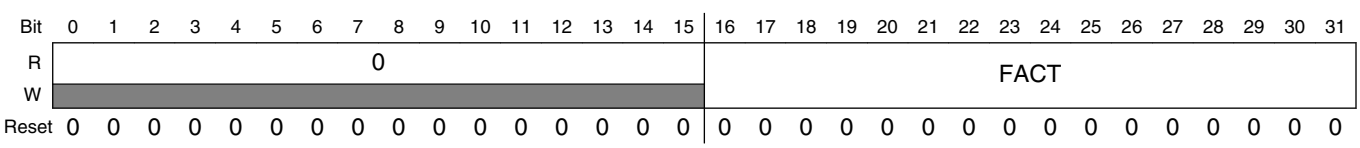
LINFlexD_BDRM field descriptions

Field	Description
0–7 DATA7	Data Byte 7 Data byte 7 of the data field.
8–15 DATA6	Data Byte 6 Data byte 6 of the data field.
16–23 DATA5	Data Byte 5 Data byte 5of the data field.
24–31 DATA4	Data Byte 4 Data byte 4 of the data field.

46.5.17 Identifier Filter Enable Register (LINFlexD_IFER)

This register enables/disables a particular filter.

Address: 0h base + 40h offset = 40h



LINFlexD_IFER field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 FACT	Filter active The software sets the bit FACT[x] to activate the filter x in Identifier list mode. Register bit can be read in any mode, written only in initialization mode.

Table continues on the next page...

LINFlexD_IFER field descriptions (continued)

Field	Description
	1. In Identifier mask mode, FACT (2n+1) have no effect on the corresponding filters as they act as mask for the Identifier 2n. 2. The length of this field depends on the value of no_of_filters. 3. The register diagram depicts the maximum no_of_filters, which can be up to 16. **NOTE:** Refer to chip configuration details to see the number of filters used in the device.

46.5.18 Identifier Filter Match Index (LINFlexD_IFMI)

This register contains the index corresponding to the received ID. It can be used to read or write the data directly in RAM.

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																											IFMI				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFMI field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 IFMI	Filter match index Upon a filter match with xth filter – IFMI[4:0] = x+1. On no match IFMI is equal to 00h. This register contains the index corresponding to the received identifier. It can be used to directly write or read the data in SRAM (see Slave mode for more details).

46.5.19 Identifier Filter Mode Register (LINFlexD_IFMR)

This register configures the modes of filters.

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																											IFM				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFMR field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 IFM	Filter mode Register bit can be read in any mode, written only in initialization mode. 0 Filters 2n and 2n+1 are in identifier list mode 1 Filters 2n and 2n+1 are in mask mode, where filter 2n+1 is the mask for filter 2n

46.5.20 Identifier Filter Control Register (LINFlexD_IFCR*n*)

This register is read-only in normal mode and can be programmed only in Initialization mode.

Even-numbered instances (IFCR0, IFCR2, IFCR4, ...) of this register act as filters in both Identifier list and Identifier mask modes.

Odd-numbered instances (IFCR1, IFCR3, IFCR5, ...) of this register act:

- As filters in Identifier list mode
- As a mask for the preceding-numbered register in identifier mask mode

Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 4Ch offset + (4d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved			DFL			DIR	CCS	0		ID					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LINFlexD_IFCR*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–18 Reserved	This field is reserved.
19–21 DFL	Data Field Length Number of data bytes in the response part of the frame. DFL = Number of data bytes - 1

Table continues on the next page...

LINFlexD_IFCR*n* field descriptions (continued)

Field	Description
	Register bit can be read in any mode, written only in initialization mode.
22 DIR	Direction This bit controls the direction of the data field. Register bit can be read in any mode, written only in initialization mode. 0 LINFlexD receives data and copies to the BDR registers 1 LINFlexD transmits data from the BDR registers
23 CCS	Classic Checksum This bit controls the type of checksum applied on the current message. Register bit can be read in any mode, written only in initialization mode. 0 Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification rev. 2.0 and higher. 1 Classic Checksum covering Data field only. This is compatible with LIN specification 1.3 and lower.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–31 ID	Identifier Identifier part of the identifier field without the identifier parity. Register bit can be read in any mode, written only in initialization mode.

46.5.21 Global Control Register (LINFlexD_GCR)

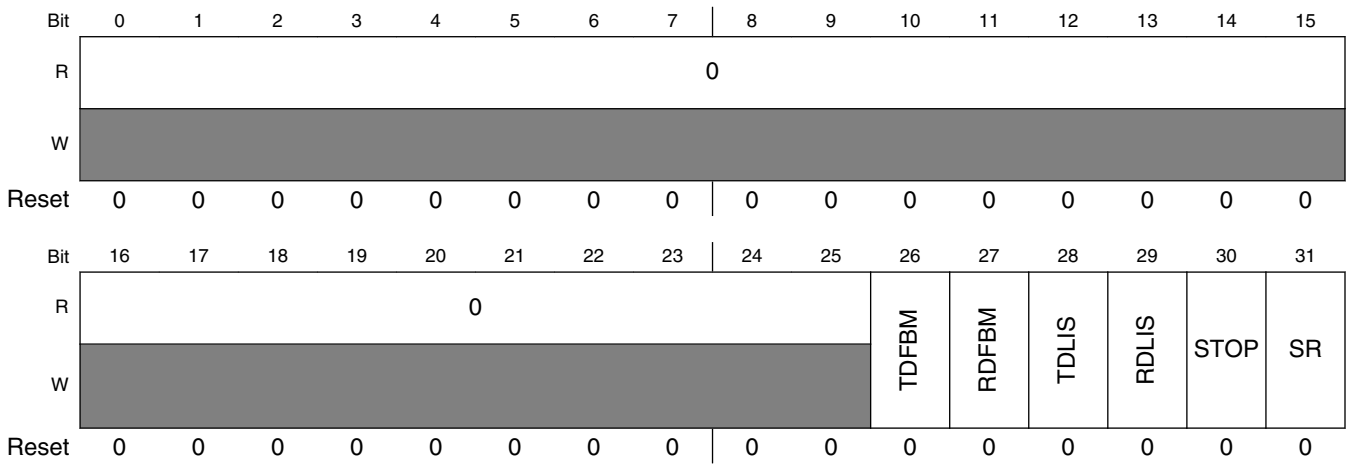
This register is read-only in Normal mode and can be programmed only in Initialization mode. This is a global control register — in other words, the register configuration will be applied in LIN mode as well as in UART mode.

The address offset depends on the no_of_filters. Refer to the chip configuration details for the number of filters used in this device.

Table 46-19. Register fields reset by SR

Register	Comment
LINSR	All fields except RXbusy and AUTOSYNC_COMP are reset
LINESR	All fields are reset
LINTCSR	Only CNT[0:7] is reset
UARTSR	All fields except RFE & TFF are reset
UARTCR	Only TFC & RFC are reset
UARTCTO	All fields are reset

Address: 0h base + 8Ch offset = 8Ch



LINFlexD_GCR field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 TDFBM	Transmit data first bit MSB This bit controls the first bit of transmit data (payload only) as MSB/LSB in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 The first bit of transmitted data is LSB — in other words, the first bit transmitted is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of transmitted data is MSB — in other words, the first bit transmitted is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
27 RDFBM	Received data first bit MSB This bit controls the first bit of received data (payload only) as MSB/LSB both in UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 The first bit of received data is LSB — in other words, the first bit received is mapped on LSB bit (BDR (0), BDR (8), BDR (16), BDR (24)) 1 The first bit of received data is MSB — in other words, the first bit received is mapped on MSB bit (BDR (7), BDR (15), BDR (23), BDR (31))
28 TDLIS	Transmit data level inversion selection This bit controls the data inversion of transmitted data (payload only) in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 Transmitted data is not inverted 1 Transmitted data is inverted
29 RDLIS	Received data level inversion selection This bit controls the data inversion of received data (payload only) in both UART and LIN modes. Register bit can be read in any mode, written only in initialization mode. 0 Received data is not inverted 1 Received data is inverted

Table continues on the next page...

LINFlexD_GCR field descriptions (continued)

Field	Description
30 STOP	1/2 stop bit configuration This bit controls the number of stop bit transmitted data in both UART and LIN modes. The stop bit is configured for all the fields (Delimiter, Sync, ID, Checksum, Payload). Register bit can be read in any mode, written only in initialization mode. 0 1 stop bit 1 2 stop bits
31 SR	Soft reset SR executes a soft reset of the LINFlexD controller (FSMs, FIFO pointers, counters, timers, status and error registers) without modifying the configuration registers when a 1 write operation is performed. This bit should be cleared by software to perform further operations (this bit is not cleared by hardware). Register bit can be written only be software in initialization mode. Bit is always read 0 by software. Table 46-19 describes the register fields reset by SR.

46.5.22 UART Preset Timeout Register (LINFlexD_UARTPTO)

This register contains the preset value of the timeout register in UART mode and is programmed according to the number of bits to be received or to monitor the idle state of the reception line. This register can be written by software any time.

The address offset depends on no_of_filters. Refer to the chip configuration details to see the number of filters used in the device.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																PTO															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

LINFlexD_UARTPTO field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 PTO	Preset Timeout PTO defines the preset value of timeout counter. A zero-value is forbidden, otherwise the UARTSR[TO] status bit is immediately set. Refer also to register UARTCTO.

46.5.23 UART Current Timeout Register (LINFlexD_UARTCTO)

This register contains the current timeout value in UART mode, and is used in conjunction with the UARTPTO register (see UART Preset Timeout Register (LINFlexD_UARTPTO) to monitor the number of bits received by UART or to monitor the idle state of the reception line. UART timeout works in both CPU and DMA modes.

The timeout counter:

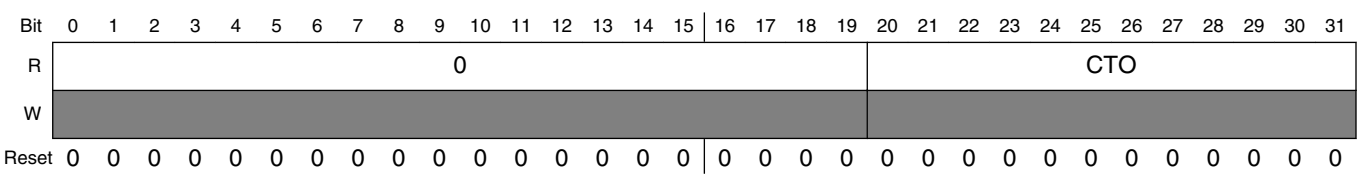
- Starts at zero and counts upward
- Is clocked with LIN_CLK / (16 * LDIV) synchronized to PBRIDGEx_CLK (when ROSE = 0)
- Is clocked with LIN_CLK / (OSR * IBRR) synchronized to PBRIDGEx_CLK (when ROSE = 1)
- Is automatically enabled when UARTCR[RXEN] = 1

It is reset when:

- Number of frames received is equal to NEF bits
- UARTCTO becomes equal to UARTPTO
- Whenever UARTPTO is written
- When DRF is set and DTU bit = 1

The address offset depends on no_of_filters. Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 94h offset = 94h



LINFlexD_UARTCTO field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–31 CTO	Current Timeout

Table continues on the next page...

LINFlexD_UARTCTO field descriptions (continued)

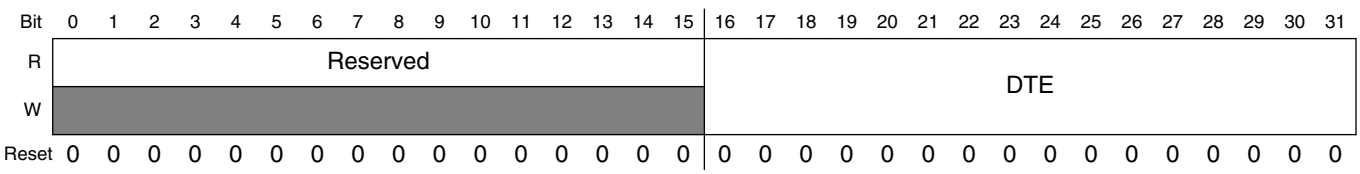
Field	Description
	CTO defines the current value of the timeout counter. CTO is a read-only field. CTO is reset every time UARTPTO is re-initialized, or UARTCTO = UARTPTO, or by hard/soft reset. When the CTO value matches the preset value (UARTPTO), the status bit UARTSR[TO] is set.

46.5.24 DMA Tx Enable Register (LINFlexD_DMATXE)

This register enables the DMA TX interface. This register can be written and read by software anytime.

The address offset depends on no_of_filters. Refer to chip configuration details to see the number of filters used in the device.

Address: 0h base + 98h offset = 98h



LINFlexD_DMATXE field descriptions

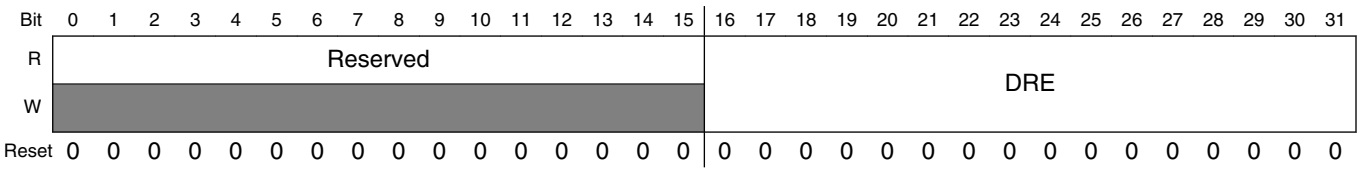
Field	Description
0–15 Reserved	This field is reserved. Reserved **NOTE:** The number of reserved bits varies, and is equal to 32 - $2^{TX_CH_NUM}$. These lies from 0 to (31 - $2^{TX_CH_NUM}$). Refer to the chip configuration details for the value of TX_CH_NUM used in this device.
16–31 DTE	DMA Tx channel Y enable **NOTE:** The actual size of the register DMATXE depends on the value of the static parameter TX_CH_NUM. The size is [$2^{TX_CH_NUM}$ - 1 : 0]. The position of these bits are (32 - $2^{TX_CH_NUM}$) to 31. When DMATXE = 0x00000000, the DMA TX interface FSM is forced (soft reset) in Idle state. **NOTE:** Refer to the chip configuration details for the value of TX_CH_NUM used in this device. 0 DMA Tx channel Y disabled 1 DMA Tx channel Y enabled

46.5.25 DMA Rx Enable Register (LINFlexD_DMARXE)

This register enables the DMA RX interface. This register can be written and read by software any time.

The address offset depends on no_of_filters. Refer to device configuration chapter to see the number of filters used in the device.

Address: 0h base + 9Ch offset = 9Ch



LINFlexD_DMARXE field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved **NOTE:** The number of reserved bits varies, and is equal to 32 - $2^{RX_CH_NUM}$. These lies from 0 to (31 - $2^{RX_CH_NUM}$). Refer to the chip configuration details for the value of RX_CH_NUM used in this device.
16–31 DRE	DMA Rx channel Y enable **NOTE:** The actual size of the register DMARXE depends on the value of the static parameter RX_CH_NUM. The size is [$2^{RX_CH_NUM}$ -1 : 0]. The position of these bits are (32 - $2^{RX_CH_NUM}$) to 31. When DMARXE = 0x00000000, the DMA RX interface FSM is forced (soft reset) in Idle state. **NOTE:** Refer to the chip configuration details for the value of RX_CH_NUM used in this device. 0 DMA Rx channel Y disabled 1 DMA Rx channel Y enabled

46.6 Programming considerations

The next subsections describe the various configurations in which the LINFlexD module can be used.

46.6.1 Master node

LINFlexD acts as Master when the MME bit is set.

46.6.1.1 Transmitter

Transmitter Master Node - Transmitter configuration.

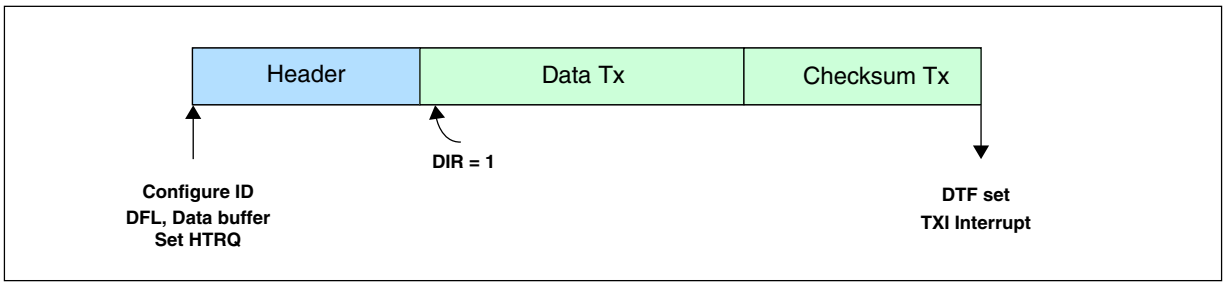


Figure 46-39. Master node – Transmitter configuration

46.6.1.2 Receiver

Receiver Master Mode - RX Configuration

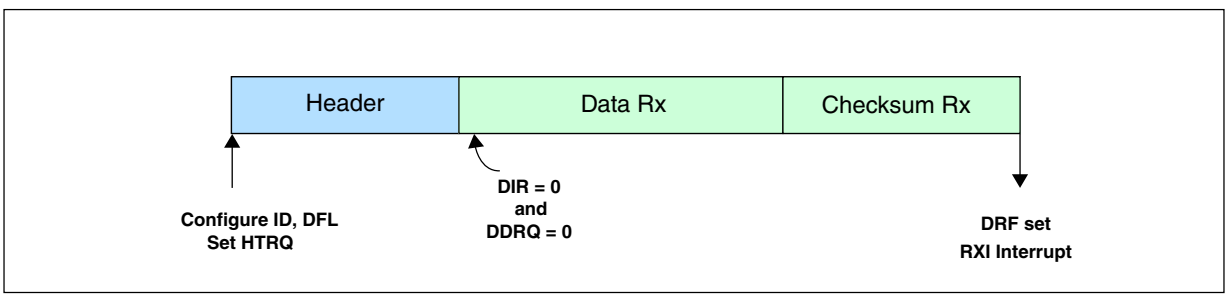


Figure 46-40. Master node – Receiver configuration

46.6.1.3 Transmitter, Bit Error

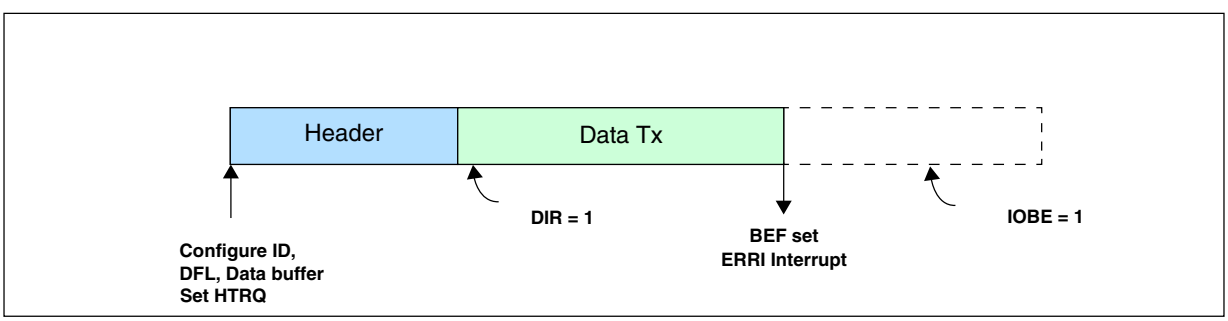


Figure 46-41. Master node – Transmitter, Bit Error configuration

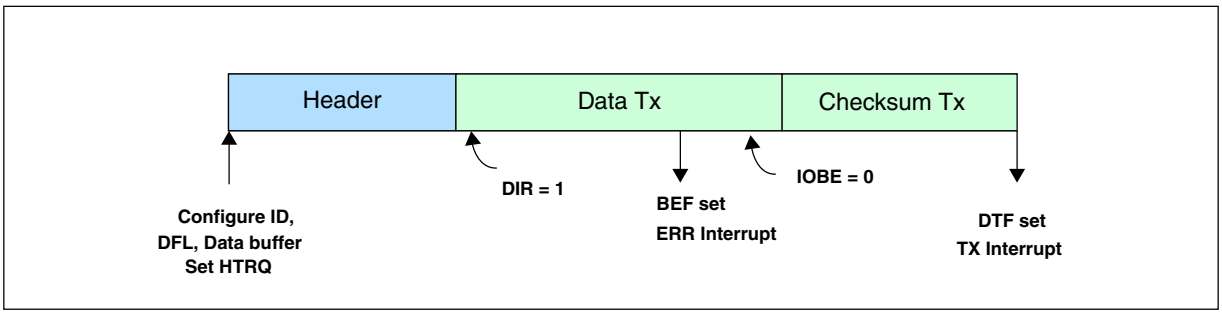


Figure 46-42. Master node – Transmitter, Checksum Error configuration

46.6.1.4 Receiver, Checksum Error

Checksum Error for Receiver.

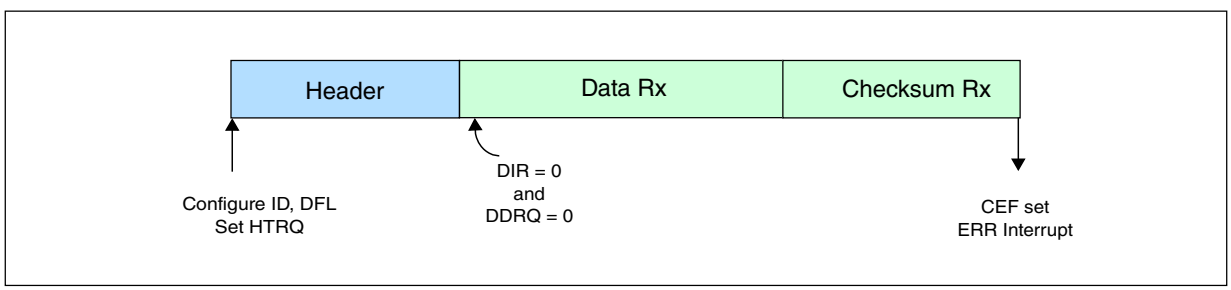


Figure 46-43. Master node – Receiver, Checksum Error configuration

46.6.2 Slave node

Slave node (transmitter).

46.6.2.1 Transmitter (no identifier filters)

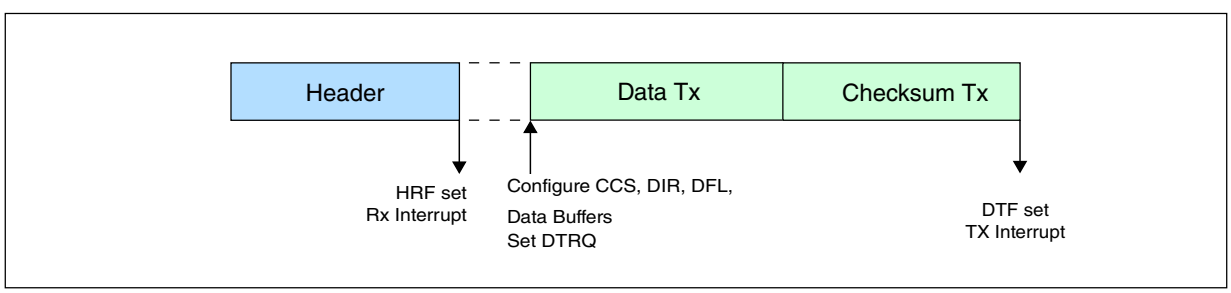


Figure 46-44. Slave node – Transmitter (no identifier filters) configuration

46.6.2.2 Receiver (no identifier filters)

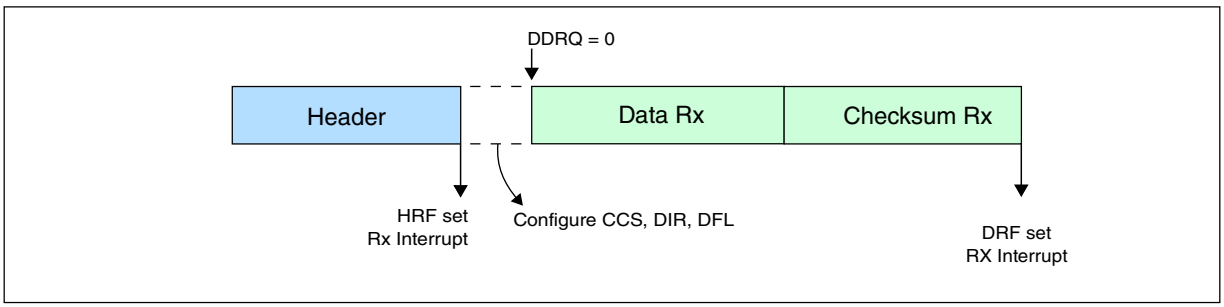


Figure 46-45. Slave node – Receiver (no identifier filters) configuration

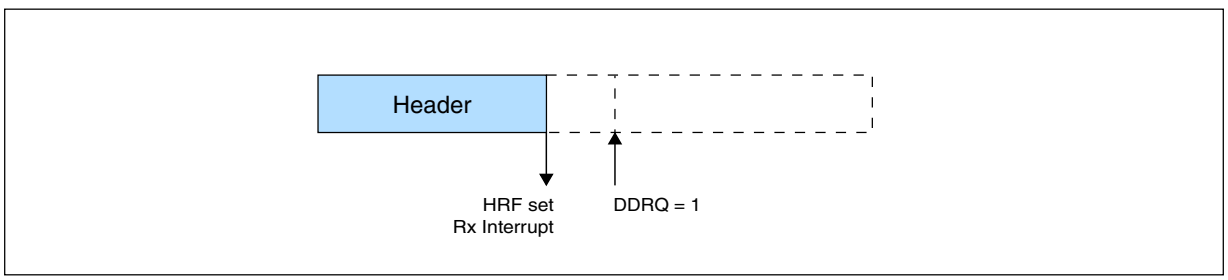


Figure 46-46. Slave node – Receiver (no identifier filters) configuration

46.6.2.3 No filters, Transmitter, Bit error

The following figure shows Slave node - No filters, Transmitter, Bit error configuration.

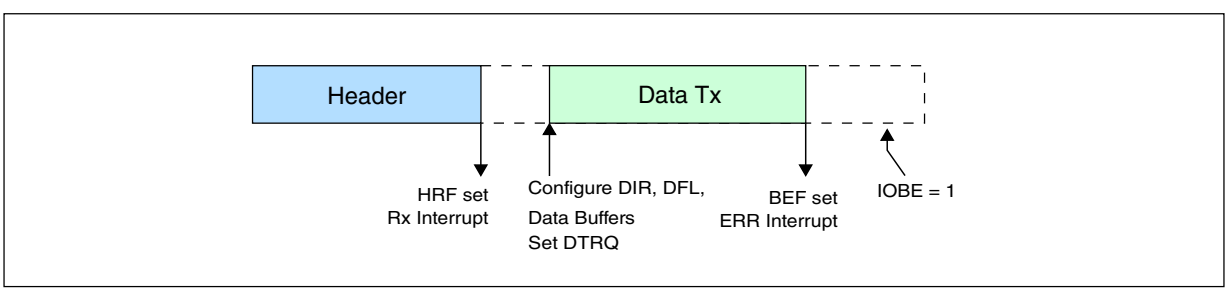


Figure 46-47. Slave node – No filters, Transmitter, Bit error configuration

46.6.2.4 No filters, Receiver, Checksum Error

The following figure shows Slave node - No filters, Receiver, Checksum error configuration.

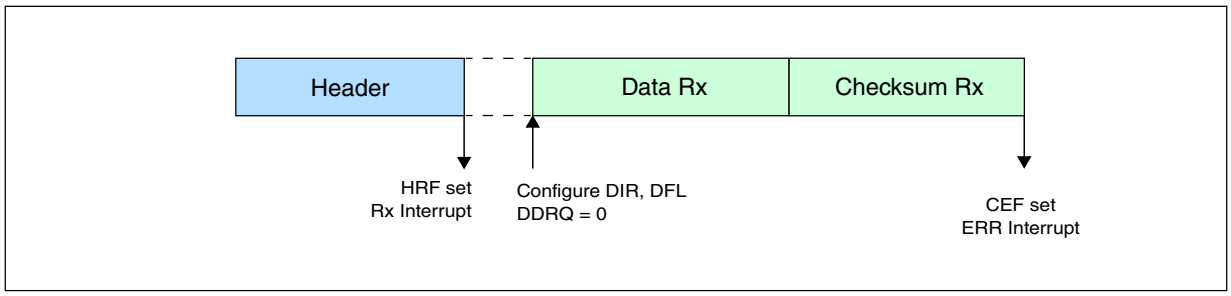


Figure 46-48. Slave node – No filters, Receiver, Checksum error configuration

46.6.2.5 At least one TX filter, BF is reset, ID matches filter

This configuration can be used in case slave never receives data, for example, sensor.

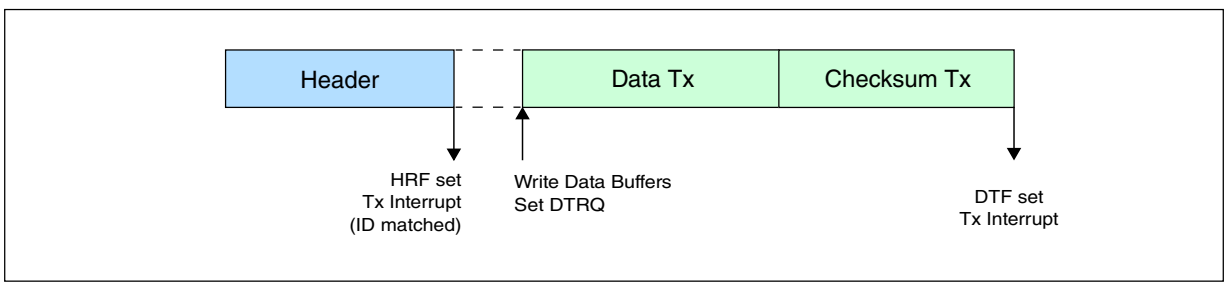


Figure 46-49. Slave node – one TX filter configuration

46.6.2.6 At least one RX filter, BF reset, ID matches filter

The following figure shows Slave node - one RX filter configuration.

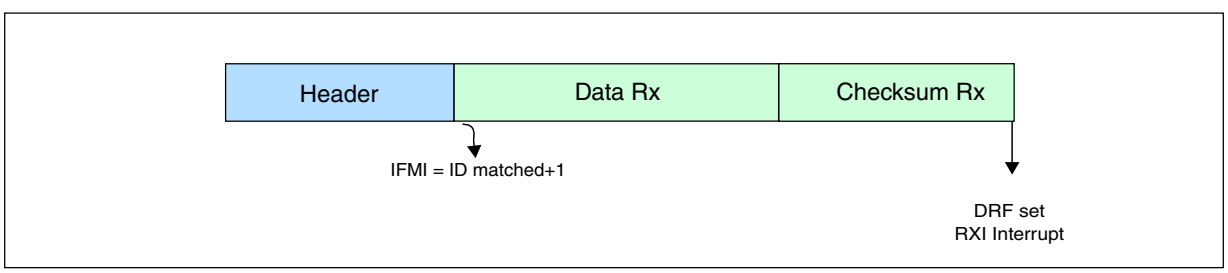


Figure 46-50. Slave node – one RX filter configuration

46.6.2.7 RX only, TX only, RX & TX filters, ID not matching filter, BF reset

The following figure shows RX only, TX only, RX and TX filters configuration.

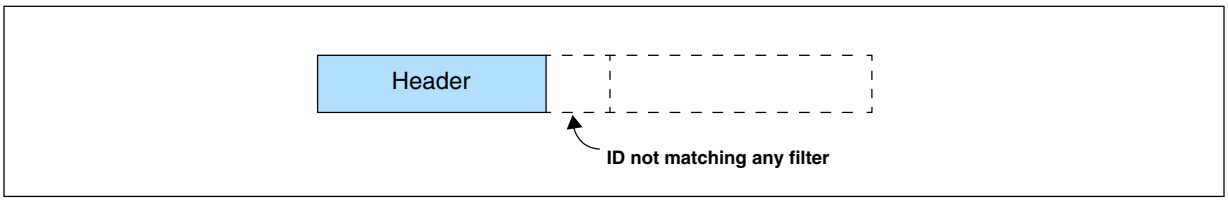


Figure 46-51. RX only, TX only, RX & TX filters configuration

46.6.2.8 TX filter, BF is set

This configuration is used when:

- All TX IDs are handled by filters.
- There are not enough other filters to handle all reception IDs.

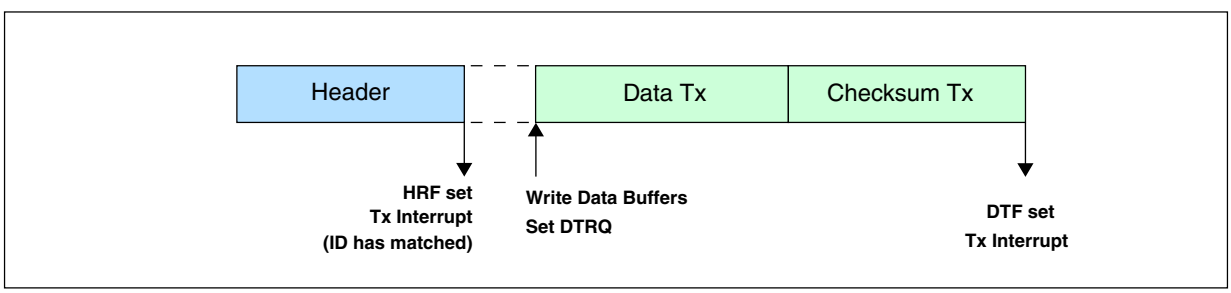


Figure 46-52. TX filter, BF is set configuration – ID has matched

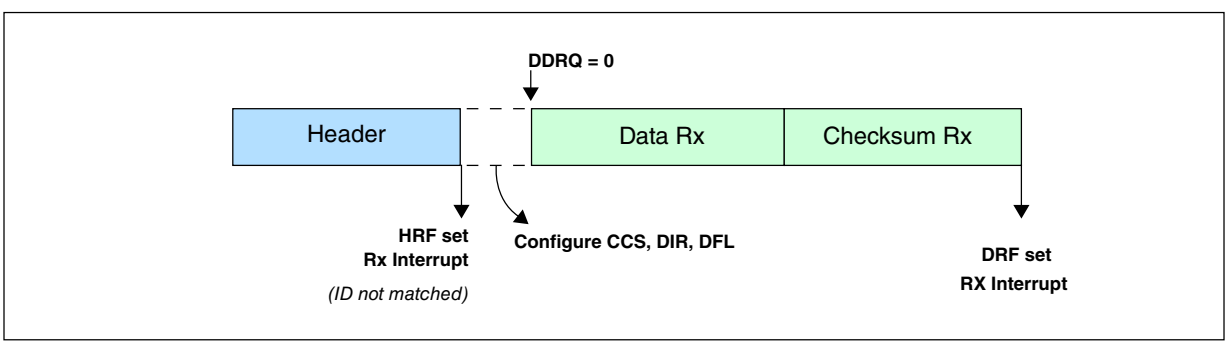


Figure 46-53. TX filter, BF is set configuration – ID not matched

46.6.2.9 RX filter, BF is set

The following figures show RX filter configurations.

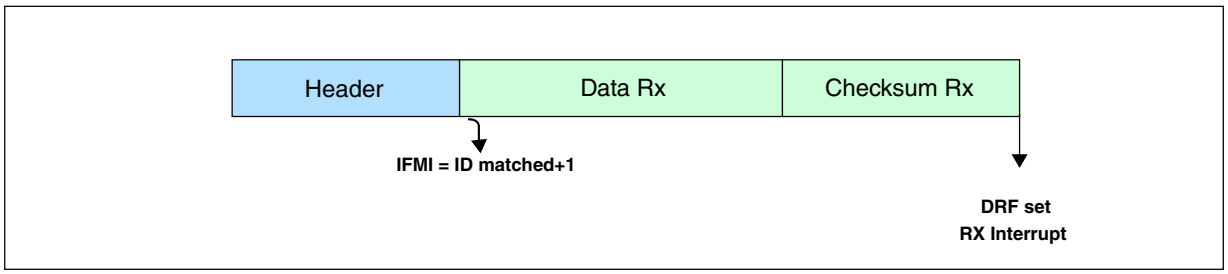


Figure 46-54. RX filter, BF is set configuration – ID matched

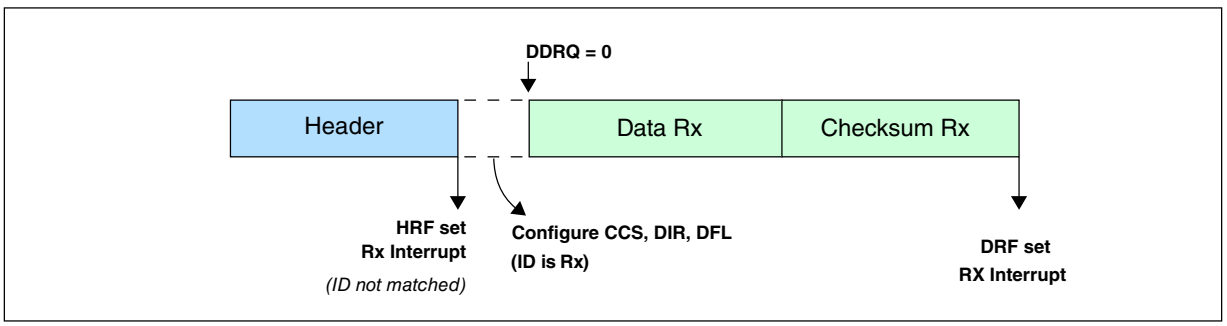


Figure 46-55. RX filter, BF is set configuration – ID not matched (ID is Rx)

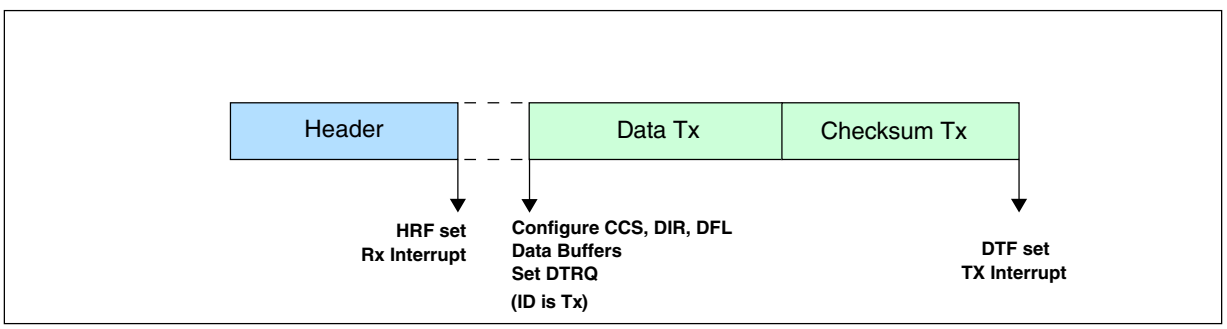


Figure 46-56. RX filter, BF is set configuration – (ID is Tx)

46.6.2.10 TX filter, RX filter, BF set

This configuration is used when:

- There are not enough filters.
- The filters are used for most frequently used IDs to reduce CPU usage.

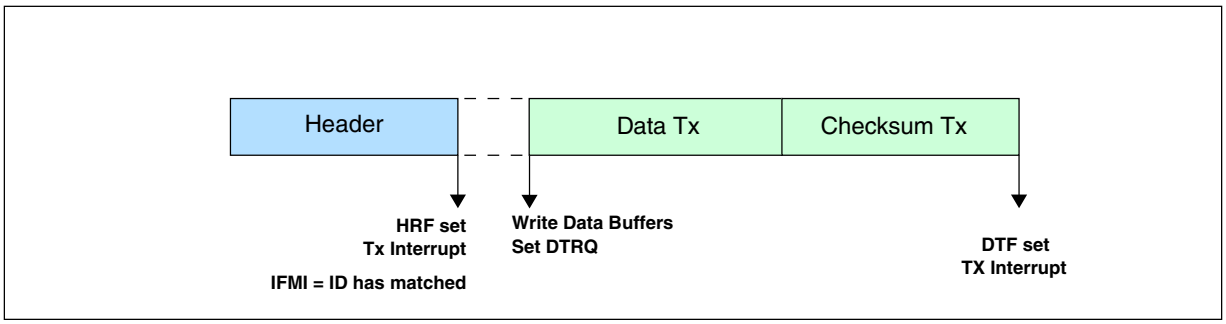


Figure 46-57. TX filter, RX filter, BF set configuration – (ID matched)

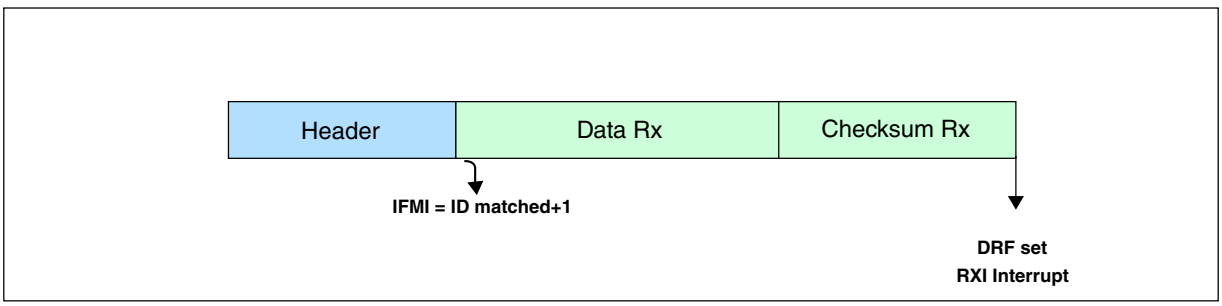


Figure 46-58. TX filter, RX filter, BF set configuration – (ID matched + 1)

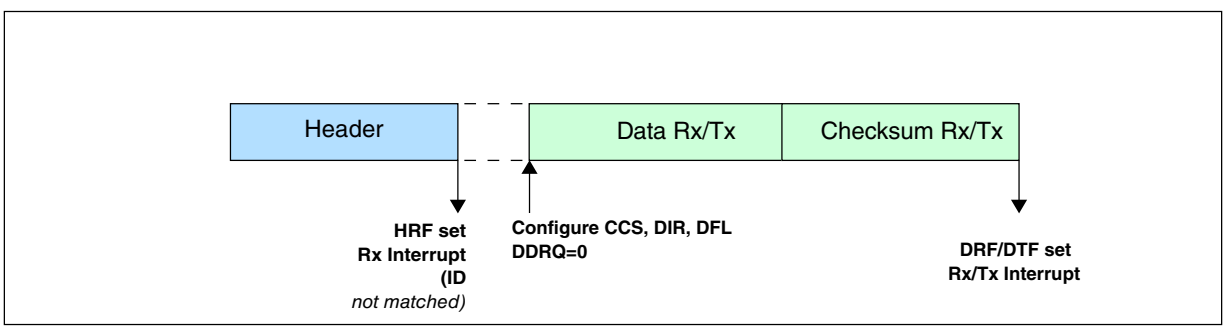


Figure 46-59. TX filter, RX filter, BF set configuration – (ID not matched, RX/TX Interrupt)

46.6.3 Timeout

Master Node: Response (during reception only) and frame timeout are checked.

Slave Node: Header, Response (during reception only), and frame timeout are checked.

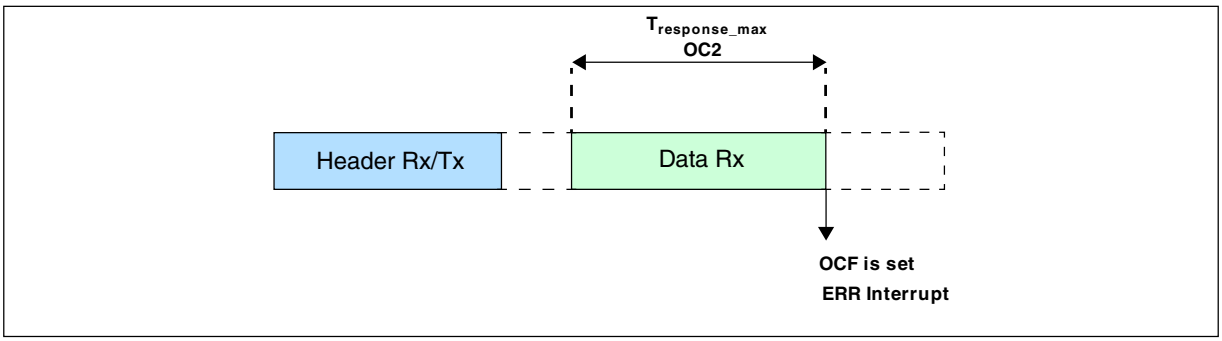


Figure 46-60. Response timeout

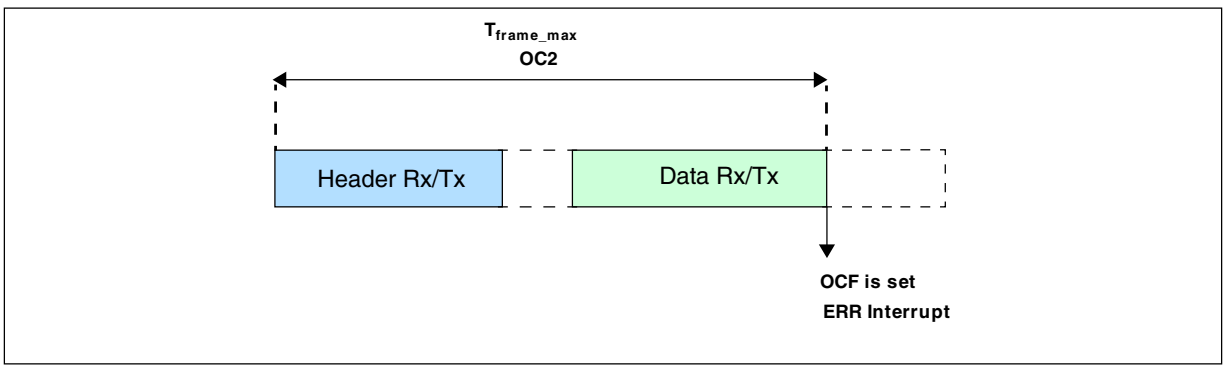


Figure 46-61. Frame timeout

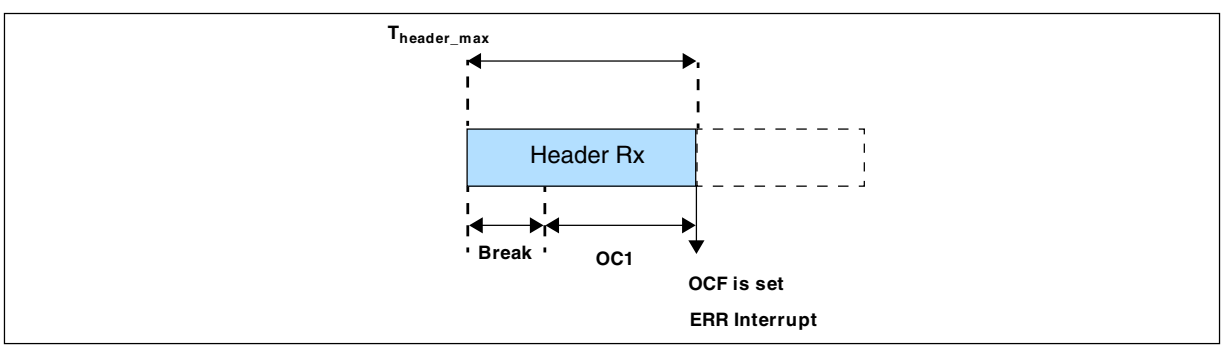


Figure 46-62. Header timeout

46.6.4 UART mode

The following figure shows UART mode configuration.

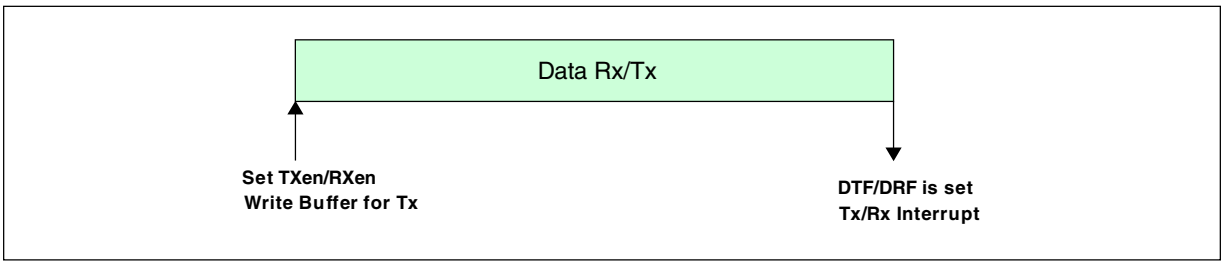


Figure 46-63. UART mode configuration

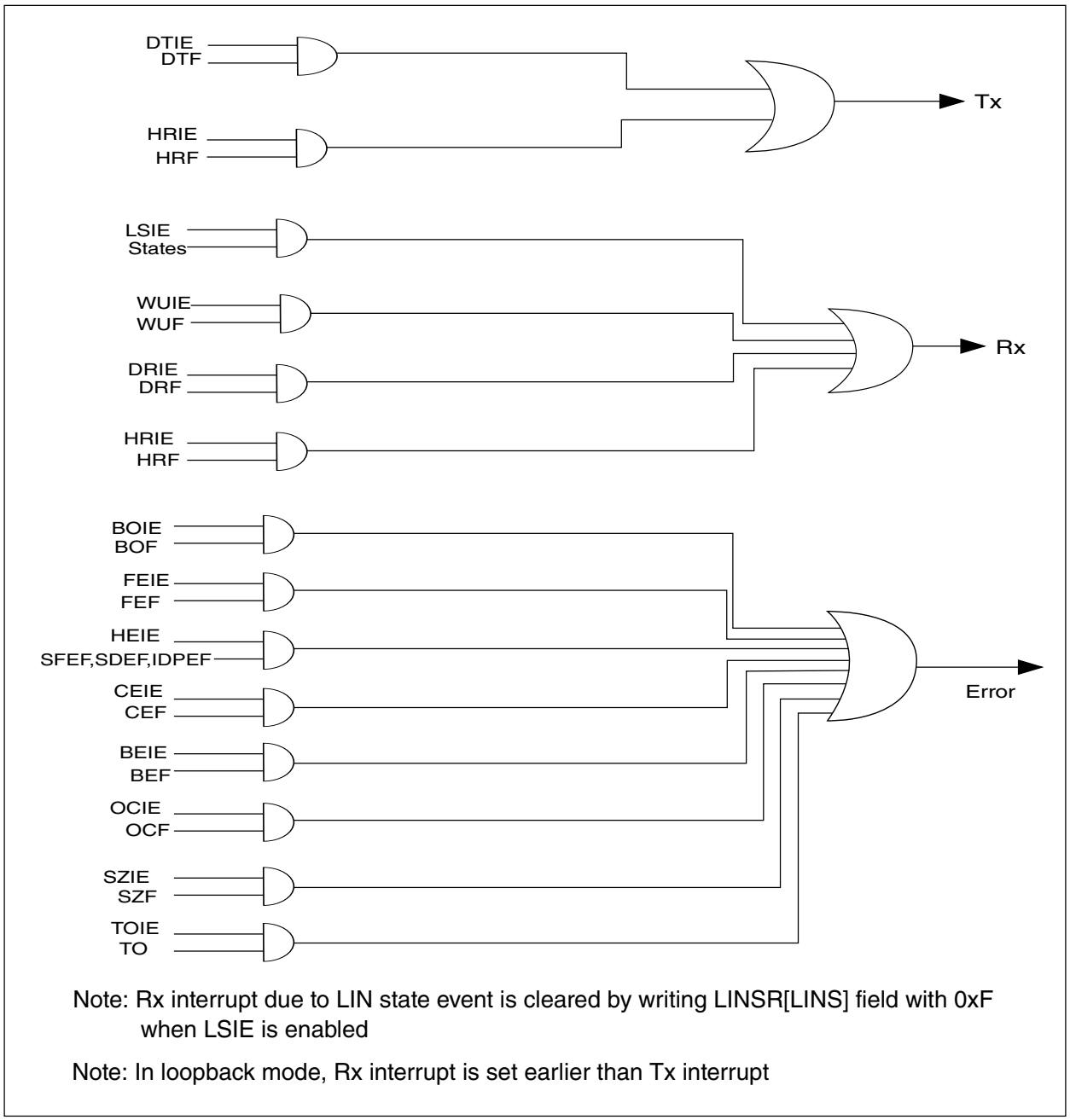
46.6.5 Interrupts

Interrupt section.

Table 46-20. Interrupts

Interrupt Event	Event flag	Enable control bit	Interrupt vector
Stuck at zero	SZF	SZIE	Error
Output compare	OCF	OCIE	Error
Bit error	BEF	BEIE	Error
Checksum error	CEF	CEIE	Error
Header error	SFEF orSDEF orIDPEF	HEIE	Error
Frame error	FEF	FEIE	Error
Buffer Overrun error	BOF	BOIE	Error
UART Timeout error	TO	TOIE	Error
Lin state	Sync Del, Sync Field,Identifier field or Checksum Field	LSIE	Rx
Wakeup	WUF	WUIE	Rx
Data Reception Complete	DRF	DRIE	Rx
Data transmitted	DTF	DTIE	Tx
Header received	HRF	HRIE	Rx
Header received	HRF	HRIE	Tx (if there is a filter match for Tx identifier)

The following figure shows an Interrupt Flow Diagram.



Note: Rx interrupt due to LIN state event is cleared by writing LINSR[LINS] field with 0xF when LSIE is enabled

Note: In loopback mode, Rx interrupt is set earlier than Tx interrupt

Figure 46-64. Interrupt diagram

46.6.6 LINFlexD Clock Tolerance

1. **Faster receiver tolerance:**
   In this case the receiver has a higher baud rate than the transmitter, thus the stop bit sampling starts already in the last transmitted payload bit. To ensure the correct noise and framing error free reception bit, the samples RS8, RS9, and RS10 must be located in the transmitted stop bit as shown in the following figure.

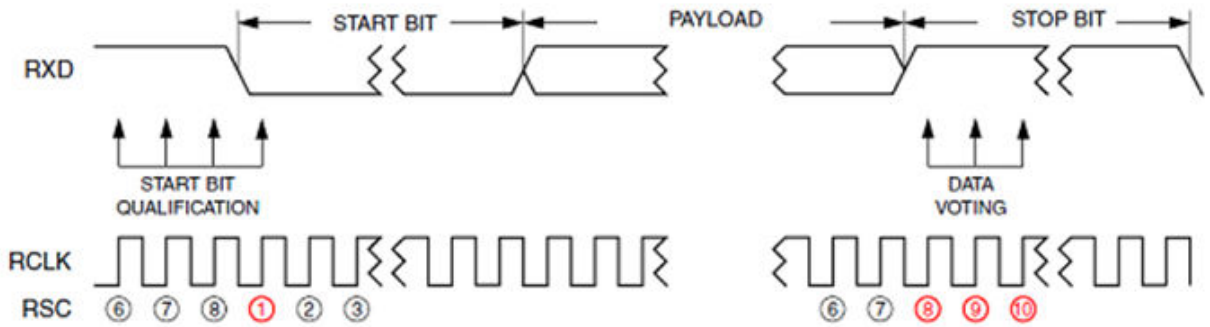


Figure 46-65. Faster receiver

2. **Slower receiver tolerance:**
   In this case the receiver has a slower baud rate than the transmitter, thus the stop bit sampling is still running while the next start bit is already transmitted. To ensure the correct noise and framing error free reception bit , the samples RS8, RS9, and RS10 must be located in the transmitted stop bit as shown in the following figure.

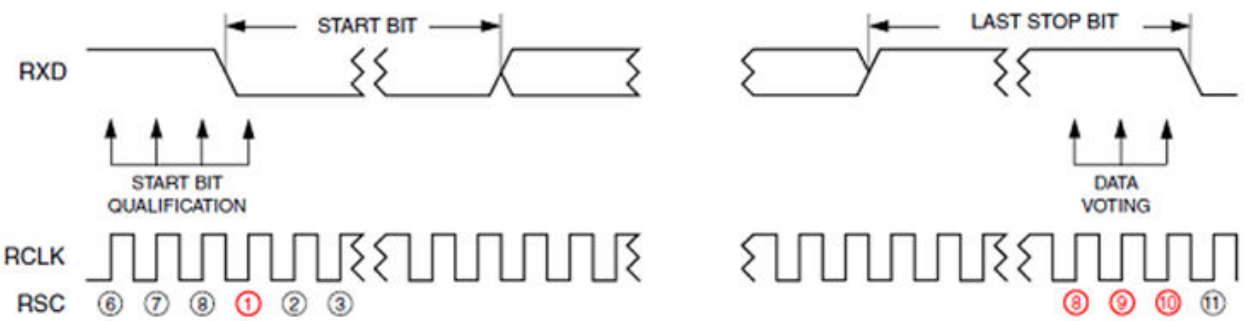


Figure 46-66. Slower receiver

Chapter 47
FlexRay

47.1 Chip-specific FlexRay information

47.1.1 FlexRay configuration

47.1.1.1 FlexRay byte swapping control

The endianess of the FlexRay AHB interface is controlled via the GPR register bits in the device.

Refer GPR Control Register (GPR_CTL) for details.

47.1.2 Stop Watch interface

For OS tick selection configuration implemented outside FlexRay stopwatch interface, see section: FlexRay OS tick selection

47.2 Introduction

This section provides a high-level summary of module features.

47.2.1 Reference

The following documents are referenced.

- *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*[1]

- Refer to this document for the all of the FlexRay related informaton including the configuration parameters and the allowed parameter ranges.
- *FlexRay Communications System Electrical Physical Layer Specification, Version 3.0*

47.2.2 Glossary

This section provides a list of terms used in this chapter.

Table 47-1. List of terms

Term	Definition
BCU	Buffer Control Unit. Handles message buffer access.
BMIF	Bus Master Interface. Provides master access to FlexRay memory area.
CC	Communication Controller
CDC	Clock Domain Crosser
CHI	Controller Host Interface
Cycle length in µT	The actual length of a cycle in µT for the ideal controller (+/- 0 ppm)
EBI	External Bus Interface
FlexRay Memory Area	Memory area to store the physical message buffer payload data, frame header, frame and slot status, and synchronization frame related tables.
FSS	Frame Start Sequence
HIF	Host Interface. Provides host access to controller.
Host	The FlexRay CC host MCU
Keyslot	Key slot is used to transmit the startup frame, sync frame, or designated single slot frame.
LUT	Look Up Table. Stores message buffer header index value.
LRAM	Look Up Table RAM. Module internal memory to store message buffer configuration data and data field offsets for individual message buffers and receive shadow buffers.
MB	Message Buffer
Mini-slot	An interval of time within the dynamic segment of the communication cycle that is of constant duration (in terms of macroticks) and that is used by the synchronized FTDMA media access scheme to manage media arbitration
MBIDX	Message Buffer Index: the position of a header field entry within the header area. If the header area is accessed as an array, this is the same as the array index of the entry.
MBNum	Message Buffer Number: Position of message buffer configuration registers within the register map. For example, Message Buffer Number 5 corresponds to the MBCCS5 register.
MCU	Microcontroller Unit
µT	Microtick
MT	Macrotick

Table continues on the next page...

1. The FlexRay Specifications have been developed for automotive applications.The FlexRay Specifications have been neither developed nor tested for non-automotive applications.

Table 47-1. List of terms (continued)

Term	Definition
MTS	Media Access Test Symbol
message frame	Frame with *Null Frame Indicator* set to 1
normal frame	null frame or message frame with both *Sync Frame Indicator* and *Startup Frame Indicator* set to 0
null frame	frame with *Null Frame Indicator* set to 0
NIT	Network Idle Time
PE	Protocol Engine
POC	Protocol Operation Control. Each state of the POC is denoted by POC:state
Rx	Reception
Slot	An interval of time during which access to a communication channel is granted exclusively to a specific node for the transmission of a frame with a frame ID corresponding to the slot.
SEQ	Sequencer Engine
SU	Status update
SECDED	Single-bit error correction, double-bit error detection
System Memory	Memory that contains the FlexRay Memory Area.
System Bus	Bus that connects the controller and System Memory
sync frame	null frame or message frame with *Sync Frame Indicator* set to 1
startup frame	null frame or message frame with both *Sync Frame Indicator* and *Startup Frame Indicator* set to 1
TCU	Time Control Unit
Tx	Transmission

47.2.3 Overview

The CC is a FlexRay communication controller that implements the FlexRay Communications System Protocol Specification, Version 2.1 Rev A.

The CC has three main components:

- *Controller host interface (CHI)*
- *Protocol engine (PE)*
- *Clock domain crossing unit (CDC)*

A block diagram of the CC with its surrounding modules is given in the below figure.

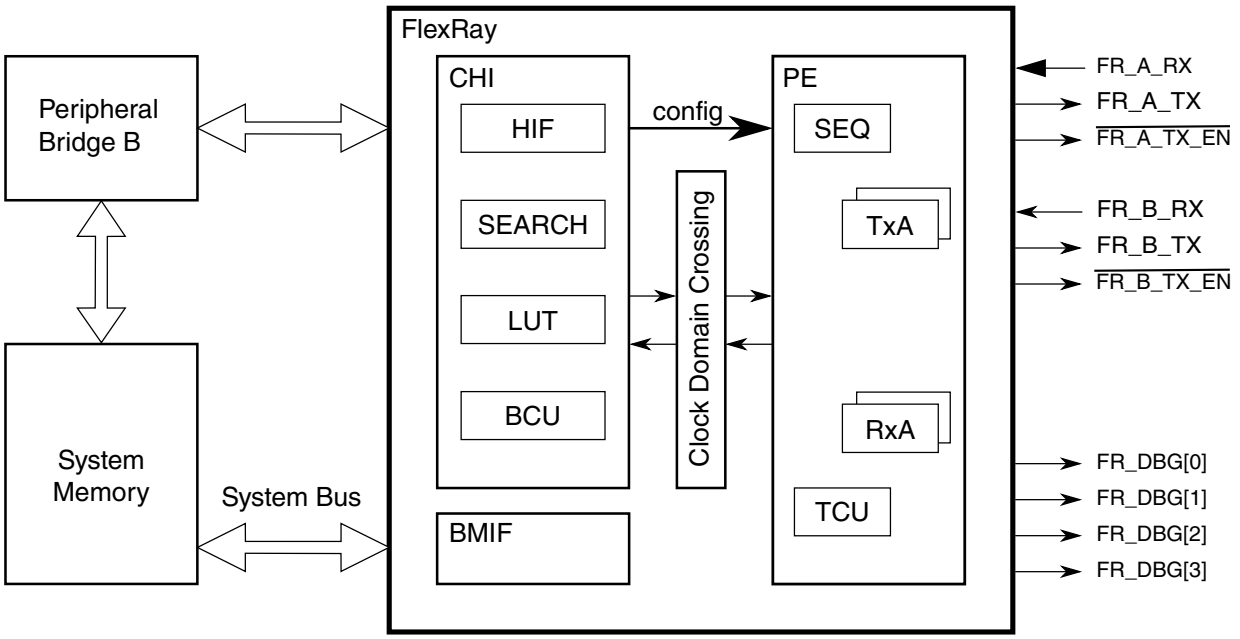


Figure 47-1. FlexRay block diagram

The protocol engine has two transmitter units TxA and TxB and two receiver units RxA and RxB for sending and receiving frames through the two FlexRay channels. The time control unit (TCU) is responsible for maintaining global clock synchronization to the FlexRay network. The overall activity of the PE is controlled by the sequencer engine (SEQ).

The controller host interface provides host access to the module's configuration, control, and status registers, as well as to the message buffer configuration, control, and status registers. The message buffers themselves, which contain the frame header and payload data received or to be transmitted, and the slot status information, are stored in the FlexRay memory area.

The clock domain crossing unit implements signal crossing from the CHI clock domain to the PE clock domain and vice versa, to allow for asynchronous PE and CHI clock domains.

The CC stores the frame header and payload data of frames received or of frames to be transmitted in the FlexRay memory area. The application accesses the FlexRay memory area to retrieve and provide the frames to be processed by the CC. In addition to the frame header and payload data, the CC stores the synchronization frame related tables in the FlexRay memory area for application processing.

The FlexRay memory area is located in the system memory of the MCU. The CC has access to the FlexRay memory area via its bus master interface (BMIF). The host provides the start address of the FlexRay memory area within the system memory by programming the System Memory Base Address Register (FR_SYMBADHR and

FR_SYMBADLR). All FlexRay memory area related offsets are stored in offset registers. The physical address pointer into the FlexRay memory area is calculated using the offset values the FlexRay memory base address.

Note

The CC does not provide a memory protection scheme for the FlexRay memory area.

47.2.4 Features

The CC provides the following features:

- *FlexRay Communications System Protocol Specification, Version 2.1 Rev A compliant protocol implementation*
- *FlexRay Communications System Electrical Physical Layer Specification, Version 3.0 compliant bus driver interface*
- Single channel support
  - FlexRay Port A can be configured to be connected either to physical FlexRay channel A or physical FlexRay channel B.
- Dual channel support
- FlexRay bus data rates of 10 Mbit/s, 8 Mbit/s, 5 Mbit/s, and 2.5 Mbit/s supported
- 128 configurable message buffers with
  - individual frame ID filtering
  - individual channel ID filtering
  - individual cycle counter filtering
- Message buffer header, status and payload data stored in dedicated FlexRay memory area
  - allows for flexible and efficient message buffer implementation
  - consistent data access ensured by means of buffer locking scheme
  - application can lock multiple buffers at the same time
- Size of message buffer payload data section configurable from 0 up to 254 bytes
- Two independent message buffer segments with configurable size of payload data section

    - each segment can contain message buffers assigned to the static segment and message buffers assigned to the dynamic segment at the same time
- Zero padding for transmit message buffers in static segment
    - applied when the frame payload length exceeds the size of the message buffer data section
- Transmit message buffers configurable with state/event semantics
- Message buffers can be configured as
    - receive message buffer
    - transmit message buffer
- Individual message buffer reconfiguration supported
    - means provided to safely disable individual message buffers
    - disabled message buffers can be reconfigured
- Two independent receive FIFOs
    - one receive FIFO per channel
    - up to 255 entries for each FIFO
    - global frame ID filtering, based on both value/mask filters and range filters
    - global channel ID filtering
    - global message ID filtering for the dynamic segment
- 4 configurable slot error counters
- 4 dedicated slot status indicators
    - used to observe slots without using receive message buffers
- Measured value indicators for the clock synchronization
    - internal synchronization frame ID and synchronization frame measurement tables can be copied into the FlexRay memory area
- Fractional macroticks are supported for clock correction
- Maskable interrupt sources provided via individual and combined interrupt lines
- 1 absolute timer
- 1 timer that can be configured to absolute or relative

- Error correction and error detection (SECDED ECC) for protocol engine data RAM
- Error detection (SECDED ECC) for CHI lookup table RAM

47.2.5 Modes of Operation

This section describes the basic operational power modes of the CC.

47.2.5.1 Disabled Mode

The CC enters the Disabled Mode during hard reset or when the host clears the ‘MEN’ field in the Module Configuration Register (FR_MCR). The host can clear this field by writing ‘0’ and only when the module is in POC:default config mode. The Disabled mode can be checked by reading the module enable field, MEN, in the Module Configuration Register (FR_MCR).

No communication is performed on the FlexRay bus.

All registers with the write access conditions *Any Time* and *Disabled Mode* can be accessed for writing as stated in the Memory Map and Register Description section.

The application configures the CC by accessing the configuration bits and fields in the Module Configuration Register (FR_MCR) as described in Module Initialization .

47.2.5.1.1 Leave Disabled Mode

The CC leaves the Disabled Mode and enters the Normal Mode, when the application writes 1 to the module enable bit MEN in the Module Configuration Register (FR_MCR)

Note

Once the CC is enabled, it can only be disabled during POC: default config.

47.2.5.2 Normal Mode

In this mode the CC is fully functional. The CC indicates that it is in Normal Mode by asserting the module enable bit MEN in the Module Configuration Register (FR_MCR).

47.2.5.2.1 Enter Normal Mode

This mode is entered when the application requests the CC to leave the Disabled Mode or when the MCU leaves the Stop Mode. If the Normal Mode was entered by leaving the Disabled Mode, the application has to perform the protocol initialization described in Protocol Initialization to achieve full FlexRay functionality.

Depending on the values of the SCM, CHA, and CHB bits in the Module Configuration Register (FR_MCR), the corresponding FlexRay bus driver ports are enabled and driven.

47.2.5.3 Stop Mode

The CC is in Stop Mode when the MCU is either in Full Stop or Pseudo Stop Mode. In this mode all CC clocks are stopped.{statement} No registers can be accessed. {statement} No communication is performed on the FlexRay Bus.

47.2.5.3.1 Enter Stop Mode

Before the application requests the MCU to enter one of the Stop Modes, it has to shut down the CC as described in Shut Down Sequence.

Note

If the CC is stopped during transmission of data, it is not guaranteed, that the FlexRay ports return to its inactive state before the clocks are stopped. This can result in the lockup of the FlexRay Bus.

47.2.5.3.2 Leave Stop Mode

The CC leaves the Stop Mode when the MCU leaves its Stop Mode and all clocks are reapplied to the CC. The CC enters the operational mode it was in before going into Stop Mode.

If the CC enters the Normal Mode, the application has to put the protocol engine into the default config state by the following sequence:

1. issue the DEFAULT_CONFIG command via Protocol Operation Control Register (FR_POCR)
2. wait for *POC:default config* in Protocol Status Register 0 (FR_PSR0)

Subsequently the application can reconfigure and/or reintegrate the FlexRay node into the FlexRay cluster.

47.3 External Signal Description

This section lists and describes the CC signals connected to external pins. These signals are summarized in the following table and described in detail in the Detailed Signal Descriptions section below.

NOTE

Please refer the device configuration of the reference manual for the exact module pin names.

NOTE

The off chip signals FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ are available on each package option. The availability of the other off-chip signals depends on the package option and is chip-specific.

Table 47-2. External Signal Properties

Name	Direction	Active	Reset	Function
FR_A_RX	Input	—	—	Receive Data Channel A
FR_A_TX	Output	—	1	Transmit Data Channel A
$\overline{\text{FR_A_TX_EN}}$	Output	Low	1	Transmit Enable Channel A
FR_B_RX	Input	—	—	Receive Data Channel B
FR_B_TX	Output	—	1	Transmit Data Channel B
$\overline{\text{FR_B_TX_EN}}$	Output	Low	1	Transmit Enable Channel B
FR_DBG[0]	Output	—	0	Debug Strobe Signal 0
FR_DBG[1]	Output	—	0	Debug Strobe Signal 1
FR_DBG[2]	Output	—	0	Debug Strobe Signal 2
FR_DBG[3]	Output	—	0	Debug Strobe Signal 3

47.3.1 Detailed Signal Descriptions

This section provides a detailed description of the CC signals, connected to external pins.

47.3.1.1 FR_A_RX — Receive Data Channel A

The FR_A_RX signal carries the receive data for channel A from the corresponding FlexRay bus driver.

47.3.1.2 FR_A_TX — Transmit Data Channel A

The FR_A_TX signal carries the transmit data for channel A to the corresponding FlexRay bus driver.

47.3.1.3 $\overline{\text{FR_A_TX_EN}}$ — Transmit Enable Channel A

The $\overline{\text{FR_A_TX_EN}}$ signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel A.

47.3.1.4 FR_B_RX — Receive Data Channel B

The FR_B_RX signal carries the receive data for channel B from the corresponding FlexRay bus driver.

47.3.1.5 FR_B_TX — Transmit Data Channel B

The FR_B_TX signal carries the transmit data for channel B to the corresponding FlexRay bus driver

47.3.1.6 $\overline{\text{FR_B_TX_EN}}$ — Transmit Enable Channel B

The $\overline{\text{FR_B_TX_EN}}$ signal indicates to the FlexRay bus driver that the CC is attempting to transmit data on channel B.

47.3.1.7 FR_DBG[3], FR_DBG[2], FR_DBG[1], FR_DBG[0] — Strobe Signals

These signals provide the selected debug strobe signals. For details on the debug strobe signal selection refer to the Strobe Signal Support.

47.4 Controller Host Interface Clocking

The clock for the CHI is derived from the system bus clock and has the same phase and frequency as the system bus clock. There are two constraints for the minimum CHI clock frequency.

The first constraint corresponds to the number of utilized message buffers and is specified in Number of Usable Message Buffers.

The second constraint corresponds to the value of the TIMEOUT field in the System Memory Access Time-Out Register (FR_SYMATOR) and is specified in Configure System Memory Access Time-Out Register (FR_SYMATOR).

47.5 Protocol Engine Clocking

The clock for the protocol engine can be generated by two sources. The first source is the internal crystal oscillator and the second source is an internal PLL. The clock source to be used is selected by the clock source select bit CLKSEL in the Module Configuration Register (FR_MCR).

Note

See the Device clocking details for the exact clock sources used.

47.5.1 Oscillator Clocking

If the protocol engine is clocked by the internal crystal oscillator, a crystal or CMOS compatible clock must be connected to the oscillator pins. The crystal or clock must fulfill the requirements given by the *FlexRay Communications System Protocol Specification, Version* 2.1 Rev A.

47.5.2 PLL Clocking

If the protocol engine is clocked by the internal PLL, the PLL clock should be divided down appropriately and provided to the PE.

For more details, see the clocking chapter of the device reference manual.

47.6 Register Descriptions

This section provides detailed descriptions of all registers in ascending address order, presented as 16-bit wide entities.

Table 47-3. Register Access Conventions

Convention	Description
	Depending on its placement in the read or write row, indicates that the bit is not readable or not writeable.
R*	Reserved bit or field, will not be changed. Application must not write any value different from the reset value.
FIELDNAME	Identifies the field. Its presence in the read or write row indicates that it can be read or written.

Table 47-4. Register Field Types

Convention	Description
rwm	A read/write bit that may be modified by a hardware in some fashion other than by a reset.
w1c	Write one to clear. A flag bit that can be read, is cleared by writing a one, writing 0 has no effect.
Reset Value	
0	Resets to zero.
1	Resets to one.
-	Not defined after reset and not affected by reset.

47.6.1 Register Reset

All registers except the Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*), Message Buffer Frame ID Register (FR_MBFIDR*n*), and Message Buffer Index Register (FR_MBIDXR*n*) are reset to their reset value on system reset. The registers mentioned above are located in physical memory blocks and, thus, they are not affected by reset. For some register fields, additional reset conditions exist. These additional reset conditions are mentioned in the detailed description of the register. The additional reset conditions are explained in the table below.

Table 47-5. Additional Register Reset Conditions

Condition	Description
Protocol RUN Command	The register field is reset when the application writes to RUN command "0101" to the POCCMD field in the Protocol Operation Control Register (FR_POCR).
Message Buffer Disable	The register field is reset when the application has disabled the message buffer. This happens when the application writes 1 to the message buffer disable trigger bit FR_MBCCSRn[EDT] while the message buffer is enabled (FR_MBCCSRn[EDS] = 1) and the CC grants the disable to the application by clearing the FR_MBCCSRn[EDS] bit.

47.6.2 Register Write Access

This section describes the write access restriction terms that apply to all registers.

47.6.2.1 Register Write Access Restriction

For each register bit and register field, the write access conditions are specified in the detailed register description. A description of the write access conditions is given in the table below. If, for a specific register bit or field, none of the given write access conditions is fulfilled, any write attempt to this register bit or field is ignored without any notification. The values of the bits or fields are not changed. The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled. The condition term [A and B] indicates that the register or field can be written to if both conditions are fulfilled.

Table 47-6. Register Write Access Restrictions

Condition	Indication	Description
Any Time	-	No write access restriction.
Disabled Mode	FR_MCR[MEN] = 0	Write access only when CC is in Disabled Mode.
Normal Mode	FR_MCR[MEN] = 1	Write access only when CC is in Normal Mode.
POC:config	FR_PSR0[PROTSTATE] = *POC:config*	Write access only when Protocol is in the *POC:config* state.
MB_DIS	FR_MBCCSR[EDS] = 0	Write access only when related Message Buffer is disabled.
MB_LCK	FR_MBCCSRn[LCKS] = 1	Write access only when related Message Buffer is locked.
IDL	FR_EEIRICR[BSY] = 0	Write access only when ECC configuration is idle.

47.6.2.2 Register Write Access Requirements

All registers can be accessed with 8-bit, 16-bit and 32-bit wide operations.

For some of the registers, at least a 16-bit wide write access is required to ensure correct operation. This write access requirement is described in the Memory map and register definition section. If an 8-bit wide write access is performed to any of these registers, this access is ignored without notification.

47.6.2.3 Internal Register Access

The following memory mapped registers are used to access multiple internal registers.

- *Strobe Signal Control Register (FR_STBSCR)*
- *Slot Status Selection Register (FR_SSSR)*
- *Slot Status Counter Condition Register (FR_SSCCR)*
- *Receive Shadow Buffer Configuration Data*

Each of these memory mapped registers provides a SEL field and a WMD bit. The SEL field is used to select the internal register. The WMD bit controls the write mode. If the WMD bit is set to 0 during the write access, all fields of the internal register are updated. If the WMD bit set to 1, only the SEL field is changed. All other fields of the internal register remain unchanged. This allows for reading back the values of the selected internal register in a subsequent read access.

47.7 Memory map and register definition

The CC occupies 8 KB (8192 bytes) of address space starting at the CC base address defined by the memory map of the MCU. Address offset - 0x0Ah should not be accessed by application as corresponding feature/s are not available. Therefore, transfer error will not be generated at these offset/s.

NOTE

The following registers are 16-bit write accessible only.

Strobe Signal Control Register (FR_STBSCR)

PE DRAM Access Register (FR_PEDRAR)

PE DRAM Data Register (FR_PEDRDR)

Sync Frame ID Rejection Filter Register (FR_SFIDRFR)

Slot Status Selection Register (FR_SSSR)

Slot Status Counter Condition Register (FR_SSCCR)

Receive Shadow Buffer Index Register (FR_RSBIR)

Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

Message Buffer Cycle Counter Filter Register (FR_MBCCFRn)

Message Buffer Frame ID Register (FR_MBFIDRn)

Message Buffer Index Register (FR_MBIDXRn)

Message Buffer Data Field Offset Register (FR_MBDORn)

LRAM ECC Error Test Register (FR_LEETRn)

FR memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Version Register (FR_MVR)	16	R	A568h	47.7.1/1875
2	Module Configuration Register (FR_MCR)	16	R/W	0000h	47.7.2/1875
4	System Memory Base Address High Register (FR_SYMBADHR)	16	R/W	0000h	47.7.3/1878
6	System Memory Base Address Low Register (FR_SYMBADLR)	16	R/W	0000h	47.7.4/1878
8	Strobe Signal Control Register (FR_STBSCR)	16	R/W	0000h	47.7.5/1879
C	Message Buffer Data Size Register (FR_MBDSR)	16	R/W	0000h	47.7.6/1881
E	Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)	16	R/W	7F7Fh	47.7.7/1881
10	PE DRAM Access Register (FR_PEDRAR)	16	R/W	0000h	47.7.8/1882
12	PE DRAM Data Register (FR_PEDRDR)	16	R/W	0000h	47.7.9/1883
14	Protocol Operation Control Register (FR_POCR)	16	R/W	0000h	47.7.10/ 1884
16	Global Interrupt Flag and Enable Register (FR_GIFER)	16	R/W	0000h	47.7.11/ 1885
18	Protocol Interrupt Flag Register 0 (FR_PIFR0)	16	R/W	0000h	47.7.12/ 1888
1A	Protocol Interrupt Flag Register 1 (FR_PIFR1)	16	R/W	0000h	47.7.13/ 1890
1C	Protocol Interrupt Enable Register 0 (FR_PIER0)	16	R/W	0000h	47.7.14/ 1892

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1E	Protocol Interrupt Enable Register 1 (FR_PIER1)	16	R/W	0000h	47.7.15/ 1894
20	CHI Error Flag Register (FR_CHIERFR)	16	R/W	0000h	47.7.16/ 1895
22	Message Buffer Interrupt Vector Register (FR_MBIVEC)	16	R	0000h	47.7.17/ 1898
24	Channel A Status Error Counter Register (FR_CASERCR)	16	R	0000h	47.7.18/ 1899
26	Channel B Status Error Counter Register (FR_CBSERCR)	16	R	0000h	47.7.19/ 1899
28	Protocol Status Register 0 (FR_PSR0)	16	R	0000h	47.7.20/ 1900
2A	Protocol Status Register 1 (FR_PSR1)	16	R	0000h	47.7.21/ 1902
2C	Protocol Status Register 2 (FR_PSR2)	16	R	0000h	47.7.22/ 1903
2E	Protocol Status Register 3 (FR_PSR3)	16	R/W	0000h	47.7.23/ 1905
30	Macrotick Counter Register (FR_MTCTR)	16	R	0000h	47.7.24/ 1907
32	Cycle Counter Register (FR_CYCTR)	16	R	0000h	47.7.25/ 1908
34	Slot Counter Channel A Register (FR_SLTCTAR)	16	R	0000h	47.7.26/ 1908
36	Slot Counter Channel B Register (FR_SLTCTBR)	16	R	0000h	47.7.27/ 1909
38	Rate Correction Value Register (FR_RTCORVR)	16	R	0000h	47.7.28/ 1909
3A	Offset Correction Value Register (FR_OFCORVR)	16	R	0000h	47.7.29/ 1910
3C	Combined Interrupt Flag Register (FR_CIFR)	16	R	0000h	47.7.30/ 1911
3E	System Memory Access Time-Out Register (FR_SYMATOR)	16	R/W	0006h	47.7.31/ 1912
40	Sync Frame Counter Register (FR_SFCNTR)	16	R	0000h	47.7.32/ 1913
42	Sync Frame Table Offset Register (FR_SFTOR)	16	R/W	0000h	47.7.33/ 1913
44	Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)	16	R/W	0000h	47.7.34/ 1914
46	Sync Frame ID Rejection Filter Register (FR_SFIDRFR)	16	R/W	0000h	47.7.35/ 1916
48	Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)	16	R/W	0000h	47.7.36/ 1916

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4A	Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)	16	R/W	0000h	47.7.37/ 1917
4C	Network Management Vector Register (FR_NMVR0)	16	R	0000h	47.7.38/ 1917
4E	Network Management Vector Register (FR_NMVR1)	16	R	0000h	47.7.38/ 1917
50	Network Management Vector Register (FR_NMVR2)	16	R	0000h	47.7.38/ 1917
52	Network Management Vector Register (FR_NMVR3)	16	R	0000h	47.7.38/ 1917
54	Network Management Vector Register (FR_NMVR4)	16	R	0000h	47.7.38/ 1917
56	Network Management Vector Register (FR_NMVR5)	16	R	0000h	47.7.38/ 1917
58	Network Management Vector Length Register (FR_NMVLR)	16	R/W	0000h	47.7.39/ 1918
5A	Timer Configuration and Control Register (FR_TICCR)	16	R/W	0000h	47.7.40/ 1918
5C	Timer 1 Cycle Set Register (FR_TI1CYSR)	16	R/W	0000h	47.7.41/ 1920
5E	Timer 1 Macrotick Offset Register (FR_TI1MTOR)	16	R/W	0000h	47.7.42/ 1921
60	Timer 2 Configuration Register 0 (Absolute Timer Configuration) (FR_TI2CR0_ABS)	16	R/W	0000h	47.7.43/ 1921
60	Timer 2 Configuration Register 0 (Relative Timer Configuration) (FR_TI2CR0_REL)	16	R/W	0000h	47.7.44/ 1922
62	Timer 2 Configuration Register 1 (Absolute Timer Configuration) (FR_TI2CR1_ABS)	16	R/W	0000h	47.7.45/ 1922
62	Timer 2 Configuration Register 1 (Relative Timer Configuration) (FR_TI2CR1_REL)	16	R/W	0000h	47.7.46/ 1923
64	Slot Status Selection Register (FR_SSSR)	16	R/W	0000h	47.7.47/ 1924
66	Slot Status Counter Condition Register (FR_SSCCR)	16	R/W	0000h	47.7.48/ 1925
68	Slot Status Register (FR_SSR0)	16	R	0000h	47.7.49/ 1927
6A	Slot Status Register (FR_SSR1)	16	R	0000h	47.7.49/ 1927
6C	Slot Status Register (FR_SSR2)	16	R	0000h	47.7.49/ 1927
6E	Slot Status Register (FR_SSR3)	16	R	0000h	47.7.49/ 1927
70	Slot Status Register (FR_SSR4)	16	R	0000h	47.7.49/ 1927

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
72	Slot Status Register (FR_SSR5)	16	R	0000h	47.7.49/ 1927
74	Slot Status Register (FR_SSR6)	16	R	0000h	47.7.49/ 1927
76	Slot Status Register (FR_SSR7)	16	R	0000h	47.7.49/ 1927
78	Slot Status Counter Register (FR_SSCR0)	16	R	0000h	47.7.50/ 1929
7A	Slot Status Counter Register (FR_SSCR1)	16	R	0000h	47.7.50/ 1929
7C	Slot Status Counter Register (FR_SSCR2)	16	R	0000h	47.7.50/ 1929
7E	Slot Status Counter Register (FR_SSCR3)	16	R	0000h	47.7.50/ 1929
80	MTS A Configuration Register (FR_MTSACFR)	16	R/W	0000h	47.7.51/ 1929
82	MTS B Configuration Register (FR_MTSBCFR)	16	R/W	0000h	47.7.52/ 1930
84	Receive Shadow Buffer Index Register (FR_RSBIR)	16	R/W	0000h	47.7.53/ 1931
86	Receive FIFO Watermark and Selection Register (FR_RFWMSR)	16	R/W	0000h	47.7.54/ 1932
88	Receive FIFO Start Index Register (FR_RFSIR)	16	R/W	0000h	47.7.55/ 1933
8A	Receive FIFO Depth and Size Register (FR_RFDSR)	16	R/W	0000h	47.7.56/ 1933
8C	Receive FIFO A Read Index Register (FR_RFARIR)	16	R	0000h	47.7.57/ 1934
8E	Receive FIFO B Read Index Register (FR_RFBRIR)	16	R	0000h	47.7.58/ 1934
90	Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)	16	R/W	0000h	47.7.59/ 1935
92	Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)	16	R/W	0000h	47.7.60/ 1935
94	Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)	16	R/W	0000h	47.7.61/ 1936
96	Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)	16	R/W	0000h	47.7.62/ 1936
98	Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)	16	R/W	0000h	47.7.63/ 1937
9A	Receive FIFO Range Filter Control Register (FR_RFRFCTR)	16	R/W	0000h	47.7.64/ 1938
9C	Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)	16	R	0000h	47.7.65/ 1939

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9E	Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)	16	R	0000h	47.7.66/ 1940
A0	Protocol Configuration Register 0 (FR_PCR0)	16	R/W	0000h	47.7.67/ 1940
A2	Protocol Configuration Register 1 (FR_PCR1)	16	R/W	0000h	47.7.68/ 1943
A4	Protocol Configuration Register 2 (FR_PCR2)	16	R/W	0000h	47.7.69/ 1943
A6	Protocol Configuration Register 3 (FR_PCR3)	16	R/W	0000h	47.7.70/ 1944
A8	Protocol Configuration Register 4 (FR_PCR4)	16	R/W	0000h	47.7.71/ 1944
AA	Protocol Configuration Register 5 (FR_PCR5)	16	R/W	0000h	47.7.72/ 1945
AC	Protocol Configuration Register 6 (FR_PCR6)	16	R/W	0000h	47.7.73/ 1945
AE	Protocol Configuration Register 7 (FR_PCR7)	16	R/W	0000h	47.7.74/ 1946
B0	Protocol Configuration Register 8 (FR_PCR8)	16	R/W	0000h	47.7.75/ 1946
B2	Protocol Configuration Register 9 (FR_PCR9)	16	R/W	0000h	47.7.76/ 1947
B4	Protocol Configuration Register 10 (FR_PCR10)	16	R/W	0000h	47.7.77/ 1948
B6	Protocol Configuration Register 11 (FR_PCR11)	16	R/W	0000h	47.7.78/ 1948
B8	Protocol Configuration Register 12 (FR_PCR12)	16	R/W	0000h	47.7.79/ 1949
BA	Protocol Configuration Register 13 (FR_PCR13)	16	R/W	0000h	47.7.80/ 1950
BC	Protocol Configuration Register 14 (FR_PCR14)	16	R/W	0000h	47.7.81/ 1950
BE	Protocol Configuration Register 15 (FR_PCR15)	16	R/W	0000h	47.7.82/ 1951
C0	Protocol Configuration Register 16 (FR_PCR16)	16	R/W	0000h	47.7.83/ 1951
C2	Protocol Configuration Register 17 (FR_PCR17)	16	R/W	0000h	47.7.84/ 1952
C4	Protocol Configuration Register 18 (FR_PCR18)	16	R/W	0000h	47.7.85/ 1952
C6	Protocol Configuration Register 19 (FR_PCR19)	16	R/W	0000h	47.7.86/ 1953
C8	Protocol Configuration Register 20 (FR_PCR20)	16	R/W	0000h	47.7.87/ 1953

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
CA	Protocol Configuration Register 21 (FR_PCR21)	16	R/W	0000h	47.7.88/ 1954
CC	Protocol Configuration Register 22 (FR_PCR22)	16	R/W	0000h	47.7.89/ 1954
CE	Protocol Configuration Register 23 (FR_PCR23)	16	R/W	0000h	47.7.90/ 1955
D0	Protocol Configuration Register 24 (FR_PCR24)	16	R/W	0000h	47.7.91/ 1955
D2	Protocol Configuration Register 25 (FR_PCR25)	16	R/W	0000h	47.7.92/ 1956
D4	Protocol Configuration Register 26 (FR_PCR26)	16	R/W	0000h	47.7.93/ 1956
D6	Protocol Configuration Register 27 (FR_PCR27)	16	R/W	0000h	47.7.94/ 1957
D8	Protocol Configuration Register 28 (FR_PCR28)	16	R/W	0000h	47.7.95/ 1958
DA	Protocol Configuration Register 29 (FR_PCR29)	16	R/W	0000h	47.7.96/ 1958
DC	Protocol Configuration Register 30 (FR_PCR30)	16	R/W	0000h	47.7.97/ 1959
DE	StopWatch Count High Register (FR_STPWHR)	16	R/W	0000h	47.7.98/ 1959
E0	Stop Watch Count Low Register (FR_STPWLR)	16	R/W	0000h	47.7.99/ 1960
E2	Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)	16	R/W	0000h	47.7.100/ 1960
E6	Receive FIFO Start Data Offset Register (FR_RFSDOR)	16	R/W	0000h	47.7.101/ 1961
E8	Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)	16	R/W	0000h	47.7.102/ 1962
EA	Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)	16	R/W	0000h	47.7.103/ 1962
EC	Receive FIFO Periodic Timer Register (FR_RFPTR)	16	R/W	0000h	47.7.104/ 1963
EE	Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)	16	R/W	0000h	47.7.105/ 1963
F0	ECC Error Interrupt Flag and Enable Register (FR_EEIFER)	16	R/W	0000h	47.7.106/ 1965
F2	ECC Error Report and Injection Control Register (FR_EERICR)	16	R/W	0000h	47.7.107/ 1967
F4	ECC Error Report Address Register (FR_EERAR)	16	R	7000h	47.7.108/ 1968
F6	ECC Error Report Data Register (FR_EERDR)	16	R	0000h	47.7.109/ 1969

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F8	ECC Error Report Code Register (FR_EERCR)	16	R	0000h	47.7.110/ 1970
FA	ECC Error Injection Address Register (FR_EEIAR)	16	R/W	0000h	47.7.111/ 1971
FC	ECC Error Injection Data Register (FR_EEIDR)	16	R/W	0000h	47.7.112/ 1971
FE	ECC Error Injection Code Register (FR_EEICR)	16	R/W	0000h	47.7.113/ 1972
800	Message Buffer Configuration, Control, Status Register (FR_MBCCSR0)	16	R/W	0000h	47.7.114/ 1972
802	Message Buffer Cycle Counter Filter Register (FR_MBCCFR0)	16	R/W	See section	47.7.115/ 1974
804	Message Buffer Frame ID Register (FR_MBFIDR0)	16	R/W	See section	47.7.116/ 1976
806	Message Buffer Index Register (FR_MBIDXR0)	16	R/W	See section	47.7.117/ 1976
808	Message Buffer Configuration, Control, Status Register (FR_MBCCSR1)	16	R/W	0000h	47.7.114/ 1972
80A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR1)	16	R/W	See section	47.7.115/ 1974
80C	Message Buffer Frame ID Register (FR_MBFIDR1)	16	R/W	See section	47.7.116/ 1976
80E	Message Buffer Index Register (FR_MBIDXR1)	16	R/W	See section	47.7.117/ 1976
810	Message Buffer Configuration, Control, Status Register (FR_MBCCSR2)	16	R/W	0000h	47.7.114/ 1972
812	Message Buffer Cycle Counter Filter Register (FR_MBCCFR2)	16	R/W	See section	47.7.115/ 1974
814	Message Buffer Frame ID Register (FR_MBFIDR2)	16	R/W	See section	47.7.116/ 1976
816	Message Buffer Index Register (FR_MBIDXR2)	16	R/W	See section	47.7.117/ 1976
818	Message Buffer Configuration, Control, Status Register (FR_MBCCSR3)	16	R/W	0000h	47.7.114/ 1972
81A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR3)	16	R/W	See section	47.7.115/ 1974
81C	Message Buffer Frame ID Register (FR_MBFIDR3)	16	R/W	See section	47.7.116/ 1976
81E	Message Buffer Index Register (FR_MBIDXR3)	16	R/W	See section	47.7.117/ 1976
820	Message Buffer Configuration, Control, Status Register (FR_MBCCSR4)	16	R/W	0000h	47.7.114/ 1972
822	Message Buffer Cycle Counter Filter Register (FR_MBCCFR4)	16	R/W	See section	47.7.115/ 1974

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
824	Message Buffer Frame ID Register (FR_MBFIDR4)	16	R/W	See section	47.7.116/ 1976
826	Message Buffer Index Register (FR_MBIDXR4)	16	R/W	See section	47.7.117/ 1976
828	Message Buffer Configuration, Control, Status Register (FR_MBCCSR5)	16	R/W	0000h	47.7.114/ 1972
82A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR5)	16	R/W	See section	47.7.115/ 1974
82C	Message Buffer Frame ID Register (FR_MBFIDR5)	16	R/W	See section	47.7.116/ 1976
82E	Message Buffer Index Register (FR_MBIDXR5)	16	R/W	See section	47.7.117/ 1976
830	Message Buffer Configuration, Control, Status Register (FR_MBCCSR6)	16	R/W	0000h	47.7.114/ 1972
832	Message Buffer Cycle Counter Filter Register (FR_MBCCFR6)	16	R/W	See section	47.7.115/ 1974
834	Message Buffer Frame ID Register (FR_MBFIDR6)	16	R/W	See section	47.7.116/ 1976
836	Message Buffer Index Register (FR_MBIDXR6)	16	R/W	See section	47.7.117/ 1976
838	Message Buffer Configuration, Control, Status Register (FR_MBCCSR7)	16	R/W	0000h	47.7.114/ 1972
83A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR7)	16	R/W	See section	47.7.115/ 1974
83C	Message Buffer Frame ID Register (FR_MBFIDR7)	16	R/W	See section	47.7.116/ 1976
83E	Message Buffer Index Register (FR_MBIDXR7)	16	R/W	See section	47.7.117/ 1976
840	Message Buffer Configuration, Control, Status Register (FR_MBCCSR8)	16	R/W	0000h	47.7.114/ 1972
842	Message Buffer Cycle Counter Filter Register (FR_MBCCFR8)	16	R/W	See section	47.7.115/ 1974
844	Message Buffer Frame ID Register (FR_MBFIDR8)	16	R/W	See section	47.7.116/ 1976
846	Message Buffer Index Register (FR_MBIDXR8)	16	R/W	See section	47.7.117/ 1976
848	Message Buffer Configuration, Control, Status Register (FR_MBCCSR9)	16	R/W	0000h	47.7.114/ 1972
84A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR9)	16	R/W	See section	47.7.115/ 1974
84C	Message Buffer Frame ID Register (FR_MBFIDR9)	16	R/W	See section	47.7.116/ 1976
84E	Message Buffer Index Register (FR_MBIDXR9)	16	R/W	See section	47.7.117/ 1976

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
850	Message Buffer Configuration, Control, Status Register (FR_MBCCSR10)	16	R/W	0000h	47.7.114/ 1972
852	Message Buffer Cycle Counter Filter Register (FR_MBCCFR10)	16	R/W	See section	47.7.115/ 1974
854	Message Buffer Frame ID Register (FR_MBFIDR10)	16	R/W	See section	47.7.116/ 1976
856	Message Buffer Index Register (FR_MBIDXR10)	16	R/W	See section	47.7.117/ 1976
858	Message Buffer Configuration, Control, Status Register (FR_MBCCSR11)	16	R/W	0000h	47.7.114/ 1972
85A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR11)	16	R/W	See section	47.7.115/ 1974
85C	Message Buffer Frame ID Register (FR_MBFIDR11)	16	R/W	See section	47.7.116/ 1976
85E	Message Buffer Index Register (FR_MBIDXR11)	16	R/W	See section	47.7.117/ 1976
860	Message Buffer Configuration, Control, Status Register (FR_MBCCSR12)	16	R/W	0000h	47.7.114/ 1972
862	Message Buffer Cycle Counter Filter Register (FR_MBCCFR12)	16	R/W	See section	47.7.115/ 1974
864	Message Buffer Frame ID Register (FR_MBFIDR12)	16	R/W	See section	47.7.116/ 1976
866	Message Buffer Index Register (FR_MBIDXR12)	16	R/W	See section	47.7.117/ 1976
868	Message Buffer Configuration, Control, Status Register (FR_MBCCSR13)	16	R/W	0000h	47.7.114/ 1972
86A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR13)	16	R/W	See section	47.7.115/ 1974
86C	Message Buffer Frame ID Register (FR_MBFIDR13)	16	R/W	See section	47.7.116/ 1976
86E	Message Buffer Index Register (FR_MBIDXR13)	16	R/W	See section	47.7.117/ 1976
870	Message Buffer Configuration, Control, Status Register (FR_MBCCSR14)	16	R/W	0000h	47.7.114/ 1972
872	Message Buffer Cycle Counter Filter Register (FR_MBCCFR14)	16	R/W	See section	47.7.115/ 1974
874	Message Buffer Frame ID Register (FR_MBFIDR14)	16	R/W	See section	47.7.116/ 1976
876	Message Buffer Index Register (FR_MBIDXR14)	16	R/W	See section	47.7.117/ 1976
878	Message Buffer Configuration, Control, Status Register (FR_MBCCSR15)	16	R/W	0000h	47.7.114/ 1972
87A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR15)	16	R/W	See section	47.7.115/ 1974

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
87C	Message Buffer Frame ID Register (FR_MBFIDR15)	16	R/W	See section	47.7.116/ 1976
87E	Message Buffer Index Register (FR_MBIDXR15)	16	R/W	See section	47.7.117/ 1976
880	Message Buffer Configuration, Control, Status Register (FR_MBCCSR16)	16	R/W	0000h	47.7.114/ 1972
882	Message Buffer Cycle Counter Filter Register (FR_MBCCFR16)	16	R/W	See section	47.7.115/ 1974
884	Message Buffer Frame ID Register (FR_MBFIDR16)	16	R/W	See section	47.7.116/ 1976
886	Message Buffer Index Register (FR_MBIDXR16)	16	R/W	See section	47.7.117/ 1976
888	Message Buffer Configuration, Control, Status Register (FR_MBCCSR17)	16	R/W	0000h	47.7.114/ 1972
88A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR17)	16	R/W	See section	47.7.115/ 1974
88C	Message Buffer Frame ID Register (FR_MBFIDR17)	16	R/W	See section	47.7.116/ 1976
88E	Message Buffer Index Register (FR_MBIDXR17)	16	R/W	See section	47.7.117/ 1976
890	Message Buffer Configuration, Control, Status Register (FR_MBCCSR18)	16	R/W	0000h	47.7.114/ 1972
892	Message Buffer Cycle Counter Filter Register (FR_MBCCFR18)	16	R/W	See section	47.7.115/ 1974
894	Message Buffer Frame ID Register (FR_MBFIDR18)	16	R/W	See section	47.7.116/ 1976
896	Message Buffer Index Register (FR_MBIDXR18)	16	R/W	See section	47.7.117/ 1976
898	Message Buffer Configuration, Control, Status Register (FR_MBCCSR19)	16	R/W	0000h	47.7.114/ 1972
89A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR19)	16	R/W	See section	47.7.115/ 1974
89C	Message Buffer Frame ID Register (FR_MBFIDR19)	16	R/W	See section	47.7.116/ 1976
89E	Message Buffer Index Register (FR_MBIDXR19)	16	R/W	See section	47.7.117/ 1976
8A0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR20)	16	R/W	0000h	47.7.114/ 1972
8A2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR20)	16	R/W	See section	47.7.115/ 1974
8A4	Message Buffer Frame ID Register (FR_MBFIDR20)	16	R/W	See section	47.7.116/ 1976
8A6	Message Buffer Index Register (FR_MBIDXR20)	16	R/W	See section	47.7.117/ 1976

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8A8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR21)	16	R/W	0000h	47.7.114/ 1972
8AA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR21)	16	R/W	See section	47.7.115/ 1974
8AC	Message Buffer Frame ID Register (FR_MBFIDR21)	16	R/W	See section	47.7.116/ 1976
8AE	Message Buffer Index Register (FR_MBIDXR21)	16	R/W	See section	47.7.117/ 1976
8B0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR22)	16	R/W	0000h	47.7.114/ 1972
8B2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR22)	16	R/W	See section	47.7.115/ 1974
8B4	Message Buffer Frame ID Register (FR_MBFIDR22)	16	R/W	See section	47.7.116/ 1976
8B6	Message Buffer Index Register (FR_MBIDXR22)	16	R/W	See section	47.7.117/ 1976
8B8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR23)	16	R/W	0000h	47.7.114/ 1972
8BA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR23)	16	R/W	See section	47.7.115/ 1974
8BC	Message Buffer Frame ID Register (FR_MBFIDR23)	16	R/W	See section	47.7.116/ 1976
8BE	Message Buffer Index Register (FR_MBIDXR23)	16	R/W	See section	47.7.117/ 1976
8C0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR24)	16	R/W	0000h	47.7.114/ 1972
8C2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR24)	16	R/W	See section	47.7.115/ 1974
8C4	Message Buffer Frame ID Register (FR_MBFIDR24)	16	R/W	See section	47.7.116/ 1976
8C6	Message Buffer Index Register (FR_MBIDXR24)	16	R/W	See section	47.7.117/ 1976
8C8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR25)	16	R/W	0000h	47.7.114/ 1972
8CA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR25)	16	R/W	See section	47.7.115/ 1974
8CC	Message Buffer Frame ID Register (FR_MBFIDR25)	16	R/W	See section	47.7.116/ 1976
8CE	Message Buffer Index Register (FR_MBIDXR25)	16	R/W	See section	47.7.117/ 1976
8D0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR26)	16	R/W	0000h	47.7.114/ 1972
8D2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR26)	16	R/W	See section	47.7.115/ 1974

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8D4	Message Buffer Frame ID Register (FR_MBFIDR26)	16	R/W	See section	47.7.116/ 1976
8D6	Message Buffer Index Register (FR_MBIDXR26)	16	R/W	See section	47.7.117/ 1976
8D8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR27)	16	R/W	0000h	47.7.114/ 1972
8DA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR27)	16	R/W	See section	47.7.115/ 1974
8DC	Message Buffer Frame ID Register (FR_MBFIDR27)	16	R/W	See section	47.7.116/ 1976
8DE	Message Buffer Index Register (FR_MBIDXR27)	16	R/W	See section	47.7.117/ 1976
8E0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR28)	16	R/W	0000h	47.7.114/ 1972
8E2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR28)	16	R/W	See section	47.7.115/ 1974
8E4	Message Buffer Frame ID Register (FR_MBFIDR28)	16	R/W	See section	47.7.116/ 1976
8E6	Message Buffer Index Register (FR_MBIDXR28)	16	R/W	See section	47.7.117/ 1976
8E8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR29)	16	R/W	0000h	47.7.114/ 1972
8EA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR29)	16	R/W	See section	47.7.115/ 1974
8EC	Message Buffer Frame ID Register (FR_MBFIDR29)	16	R/W	See section	47.7.116/ 1976
8EE	Message Buffer Index Register (FR_MBIDXR29)	16	R/W	See section	47.7.117/ 1976
8F0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR30)	16	R/W	0000h	47.7.114/ 1972
8F2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR30)	16	R/W	See section	47.7.115/ 1974
8F4	Message Buffer Frame ID Register (FR_MBFIDR30)	16	R/W	See section	47.7.116/ 1976
8F6	Message Buffer Index Register (FR_MBIDXR30)	16	R/W	See section	47.7.117/ 1976
8F8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR31)	16	R/W	0000h	47.7.114/ 1972
8FA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR31)	16	R/W	See section	47.7.115/ 1974
8FC	Message Buffer Frame ID Register (FR_MBFIDR31)	16	R/W	See section	47.7.116/ 1976
8FE	Message Buffer Index Register (FR_MBIDXR31)	16	R/W	See section	47.7.117/ 1976

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
900	Message Buffer Configuration, Control, Status Register (FR_MBCCSR32)	16	R/W	0000h	47.7.114/ 1972
902	Message Buffer Cycle Counter Filter Register (FR_MBCCFR32)	16	R/W	See section	47.7.115/ 1974
904	Message Buffer Frame ID Register (FR_MBFIDR32)	16	R/W	See section	47.7.116/ 1976
906	Message Buffer Index Register (FR_MBIDXR32)	16	R/W	See section	47.7.117/ 1976
908	Message Buffer Configuration, Control, Status Register (FR_MBCCSR33)	16	R/W	0000h	47.7.114/ 1972
90A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR33)	16	R/W	See section	47.7.115/ 1974
90C	Message Buffer Frame ID Register (FR_MBFIDR33)	16	R/W	See section	47.7.116/ 1976
90E	Message Buffer Index Register (FR_MBIDXR33)	16	R/W	See section	47.7.117/ 1976
910	Message Buffer Configuration, Control, Status Register (FR_MBCCSR34)	16	R/W	0000h	47.7.114/ 1972
912	Message Buffer Cycle Counter Filter Register (FR_MBCCFR34)	16	R/W	See section	47.7.115/ 1974
914	Message Buffer Frame ID Register (FR_MBFIDR34)	16	R/W	See section	47.7.116/ 1976
916	Message Buffer Index Register (FR_MBIDXR34)	16	R/W	See section	47.7.117/ 1976
918	Message Buffer Configuration, Control, Status Register (FR_MBCCSR35)	16	R/W	0000h	47.7.114/ 1972
91A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR35)	16	R/W	See section	47.7.115/ 1974
91C	Message Buffer Frame ID Register (FR_MBFIDR35)	16	R/W	See section	47.7.116/ 1976
91E	Message Buffer Index Register (FR_MBIDXR35)	16	R/W	See section	47.7.117/ 1976
920	Message Buffer Configuration, Control, Status Register (FR_MBCCSR36)	16	R/W	0000h	47.7.114/ 1972
922	Message Buffer Cycle Counter Filter Register (FR_MBCCFR36)	16	R/W	See section	47.7.115/ 1974
924	Message Buffer Frame ID Register (FR_MBFIDR36)	16	R/W	See section	47.7.116/ 1976
926	Message Buffer Index Register (FR_MBIDXR36)	16	R/W	See section	47.7.117/ 1976
928	Message Buffer Configuration, Control, Status Register (FR_MBCCSR37)	16	R/W	0000h	47.7.114/ 1972
92A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR37)	16	R/W	See section	47.7.115/ 1974

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
92C	Message Buffer Frame ID Register (FR_MBFIDR37)	16	R/W	See section	47.7.116/ 1976
92E	Message Buffer Index Register (FR_MBIDXR37)	16	R/W	See section	47.7.117/ 1976
930	Message Buffer Configuration, Control, Status Register (FR_MBCCSR38)	16	R/W	0000h	47.7.114/ 1972
932	Message Buffer Cycle Counter Filter Register (FR_MBCCFR38)	16	R/W	See section	47.7.115/ 1974
934	Message Buffer Frame ID Register (FR_MBFIDR38)	16	R/W	See section	47.7.116/ 1976
936	Message Buffer Index Register (FR_MBIDXR38)	16	R/W	See section	47.7.117/ 1976
938	Message Buffer Configuration, Control, Status Register (FR_MBCCSR39)	16	R/W	0000h	47.7.114/ 1972
93A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR39)	16	R/W	See section	47.7.115/ 1974
93C	Message Buffer Frame ID Register (FR_MBFIDR39)	16	R/W	See section	47.7.116/ 1976
93E	Message Buffer Index Register (FR_MBIDXR39)	16	R/W	See section	47.7.117/ 1976
940	Message Buffer Configuration, Control, Status Register (FR_MBCCSR40)	16	R/W	0000h	47.7.114/ 1972
942	Message Buffer Cycle Counter Filter Register (FR_MBCCFR40)	16	R/W	See section	47.7.115/ 1974
944	Message Buffer Frame ID Register (FR_MBFIDR40)	16	R/W	See section	47.7.116/ 1976
946	Message Buffer Index Register (FR_MBIDXR40)	16	R/W	See section	47.7.117/ 1976
948	Message Buffer Configuration, Control, Status Register (FR_MBCCSR41)	16	R/W	0000h	47.7.114/ 1972
94A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR41)	16	R/W	See section	47.7.115/ 1974
94C	Message Buffer Frame ID Register (FR_MBFIDR41)	16	R/W	See section	47.7.116/ 1976
94E	Message Buffer Index Register (FR_MBIDXR41)	16	R/W	See section	47.7.117/ 1976
950	Message Buffer Configuration, Control, Status Register (FR_MBCCSR42)	16	R/W	0000h	47.7.114/ 1972
952	Message Buffer Cycle Counter Filter Register (FR_MBCCFR42)	16	R/W	See section	47.7.115/ 1974
954	Message Buffer Frame ID Register (FR_MBFIDR42)	16	R/W	See section	47.7.116/ 1976
956	Message Buffer Index Register (FR_MBIDXR42)	16	R/W	See section	47.7.117/ 1976

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
958	Message Buffer Configuration, Control, Status Register (FR_MBCCSR43)	16	R/W	0000h	47.7.114/ 1972
95A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR43)	16	R/W	See section	47.7.115/ 1974
95C	Message Buffer Frame ID Register (FR_MBFIDR43)	16	R/W	See section	47.7.116/ 1976
95E	Message Buffer Index Register (FR_MBIDXR43)	16	R/W	See section	47.7.117/ 1976
960	Message Buffer Configuration, Control, Status Register (FR_MBCCSR44)	16	R/W	0000h	47.7.114/ 1972
962	Message Buffer Cycle Counter Filter Register (FR_MBCCFR44)	16	R/W	See section	47.7.115/ 1974
964	Message Buffer Frame ID Register (FR_MBFIDR44)	16	R/W	See section	47.7.116/ 1976
966	Message Buffer Index Register (FR_MBIDXR44)	16	R/W	See section	47.7.117/ 1976
968	Message Buffer Configuration, Control, Status Register (FR_MBCCSR45)	16	R/W	0000h	47.7.114/ 1972
96A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR45)	16	R/W	See section	47.7.115/ 1974
96C	Message Buffer Frame ID Register (FR_MBFIDR45)	16	R/W	See section	47.7.116/ 1976
96E	Message Buffer Index Register (FR_MBIDXR45)	16	R/W	See section	47.7.117/ 1976
970	Message Buffer Configuration, Control, Status Register (FR_MBCCSR46)	16	R/W	0000h	47.7.114/ 1972
972	Message Buffer Cycle Counter Filter Register (FR_MBCCFR46)	16	R/W	See section	47.7.115/ 1974
974	Message Buffer Frame ID Register (FR_MBFIDR46)	16	R/W	See section	47.7.116/ 1976
976	Message Buffer Index Register (FR_MBIDXR46)	16	R/W	See section	47.7.117/ 1976
978	Message Buffer Configuration, Control, Status Register (FR_MBCCSR47)	16	R/W	0000h	47.7.114/ 1972
97A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR47)	16	R/W	See section	47.7.115/ 1974
97C	Message Buffer Frame ID Register (FR_MBFIDR47)	16	R/W	See section	47.7.116/ 1976
97E	Message Buffer Index Register (FR_MBIDXR47)	16	R/W	See section	47.7.117/ 1976
980	Message Buffer Configuration, Control, Status Register (FR_MBCCSR48)	16	R/W	0000h	47.7.114/ 1972
982	Message Buffer Cycle Counter Filter Register (FR_MBCCFR48)	16	R/W	See section	47.7.115/ 1974

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
984	Message Buffer Frame ID Register (FR_MBFIDR48)	16	R/W	See section	47.7.116/ 1976
986	Message Buffer Index Register (FR_MBIDXR48)	16	R/W	See section	47.7.117/ 1976
988	Message Buffer Configuration, Control, Status Register (FR_MBCCSR49)	16	R/W	0000h	47.7.114/ 1972
98A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR49)	16	R/W	See section	47.7.115/ 1974
98C	Message Buffer Frame ID Register (FR_MBFIDR49)	16	R/W	See section	47.7.116/ 1976
98E	Message Buffer Index Register (FR_MBIDXR49)	16	R/W	See section	47.7.117/ 1976
990	Message Buffer Configuration, Control, Status Register (FR_MBCCSR50)	16	R/W	0000h	47.7.114/ 1972
992	Message Buffer Cycle Counter Filter Register (FR_MBCCFR50)	16	R/W	See section	47.7.115/ 1974
994	Message Buffer Frame ID Register (FR_MBFIDR50)	16	R/W	See section	47.7.116/ 1976
996	Message Buffer Index Register (FR_MBIDXR50)	16	R/W	See section	47.7.117/ 1976
998	Message Buffer Configuration, Control, Status Register (FR_MBCCSR51)	16	R/W	0000h	47.7.114/ 1972
99A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR51)	16	R/W	See section	47.7.115/ 1974
99C	Message Buffer Frame ID Register (FR_MBFIDR51)	16	R/W	See section	47.7.116/ 1976
99E	Message Buffer Index Register (FR_MBIDXR51)	16	R/W	See section	47.7.117/ 1976
9A0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR52)	16	R/W	0000h	47.7.114/ 1972
9A2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR52)	16	R/W	See section	47.7.115/ 1974
9A4	Message Buffer Frame ID Register (FR_MBFIDR52)	16	R/W	See section	47.7.116/ 1976
9A6	Message Buffer Index Register (FR_MBIDXR52)	16	R/W	See section	47.7.117/ 1976
9A8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR53)	16	R/W	0000h	47.7.114/ 1972
9AA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR53)	16	R/W	See section	47.7.115/ 1974
9AC	Message Buffer Frame ID Register (FR_MBFIDR53)	16	R/W	See section	47.7.116/ 1976
9AE	Message Buffer Index Register (FR_MBIDXR53)	16	R/W	See section	47.7.117/ 1976

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9B0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR54)	16	R/W	0000h	47.7.114/ 1972
9B2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR54)	16	R/W	See section	47.7.115/ 1974
9B4	Message Buffer Frame ID Register (FR_MBFIDR54)	16	R/W	See section	47.7.116/ 1976
9B6	Message Buffer Index Register (FR_MBIDXR54)	16	R/W	See section	47.7.117/ 1976
9B8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR55)	16	R/W	0000h	47.7.114/ 1972
9BA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR55)	16	R/W	See section	47.7.115/ 1974
9BC	Message Buffer Frame ID Register (FR_MBFIDR55)	16	R/W	See section	47.7.116/ 1976
9BE	Message Buffer Index Register (FR_MBIDXR55)	16	R/W	See section	47.7.117/ 1976
9C0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR56)	16	R/W	0000h	47.7.114/ 1972
9C2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR56)	16	R/W	See section	47.7.115/ 1974
9C4	Message Buffer Frame ID Register (FR_MBFIDR56)	16	R/W	See section	47.7.116/ 1976
9C6	Message Buffer Index Register (FR_MBIDXR56)	16	R/W	See section	47.7.117/ 1976
9C8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR57)	16	R/W	0000h	47.7.114/ 1972
9CA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR57)	16	R/W	See section	47.7.115/ 1974
9CC	Message Buffer Frame ID Register (FR_MBFIDR57)	16	R/W	See section	47.7.116/ 1976
9CE	Message Buffer Index Register (FR_MBIDXR57)	16	R/W	See section	47.7.117/ 1976
9D0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR58)	16	R/W	0000h	47.7.114/ 1972
9D2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR58)	16	R/W	See section	47.7.115/ 1974
9D4	Message Buffer Frame ID Register (FR_MBFIDR58)	16	R/W	See section	47.7.116/ 1976
9D6	Message Buffer Index Register (FR_MBIDXR58)	16	R/W	See section	47.7.117/ 1976
9D8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR59)	16	R/W	0000h	47.7.114/ 1972
9DA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR59)	16	R/W	See section	47.7.115/ 1974

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9DC	Message Buffer Frame ID Register (FR_MBFIDR59)	16	R/W	See section	47.7.116/ 1976
9DE	Message Buffer Index Register (FR_MBIDXR59)	16	R/W	See section	47.7.117/ 1976
9E0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR60)	16	R/W	0000h	47.7.114/ 1972
9E2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR60)	16	R/W	See section	47.7.115/ 1974
9E4	Message Buffer Frame ID Register (FR_MBFIDR60)	16	R/W	See section	47.7.116/ 1976
9E6	Message Buffer Index Register (FR_MBIDXR60)	16	R/W	See section	47.7.117/ 1976
9E8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR61)	16	R/W	0000h	47.7.114/ 1972
9EA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR61)	16	R/W	See section	47.7.115/ 1974
9EC	Message Buffer Frame ID Register (FR_MBFIDR61)	16	R/W	See section	47.7.116/ 1976
9EE	Message Buffer Index Register (FR_MBIDXR61)	16	R/W	See section	47.7.117/ 1976
9F0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR62)	16	R/W	0000h	47.7.114/ 1972
9F2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR62)	16	R/W	See section	47.7.115/ 1974
9F4	Message Buffer Frame ID Register (FR_MBFIDR62)	16	R/W	See section	47.7.116/ 1976
9F6	Message Buffer Index Register (FR_MBIDXR62)	16	R/W	See section	47.7.117/ 1976
9F8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR63)	16	R/W	0000h	47.7.114/ 1972
9FA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR63)	16	R/W	See section	47.7.115/ 1974
9FC	Message Buffer Frame ID Register (FR_MBFIDR63)	16	R/W	See section	47.7.116/ 1976
9FE	Message Buffer Index Register (FR_MBIDXR63)	16	R/W	See section	47.7.117/ 1976
A00	Message Buffer Configuration, Control, Status Register (FR_MBCCSR64)	16	R/W	0000h	47.7.114/ 1972
A02	Message Buffer Cycle Counter Filter Register (FR_MBCCFR64)	16	R/W	See section	47.7.115/ 1974
A04	Message Buffer Frame ID Register (FR_MBFIDR64)	16	R/W	See section	47.7.116/ 1976
A06	Message Buffer Index Register (FR_MBIDXR64)	16	R/W	See section	47.7.117/ 1976

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A08	Message Buffer Configuration, Control, Status Register (FR_MBCCSR65)	16	R/W	0000h	47.7.114/ 1972
A0A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR65)	16	R/W	See section	47.7.115/ 1974
A0C	Message Buffer Frame ID Register (FR_MBFIDR65)	16	R/W	See section	47.7.116/ 1976
A0E	Message Buffer Index Register (FR_MBIDXR65)	16	R/W	See section	47.7.117/ 1976
A10	Message Buffer Configuration, Control, Status Register (FR_MBCCSR66)	16	R/W	0000h	47.7.114/ 1972
A12	Message Buffer Cycle Counter Filter Register (FR_MBCCFR66)	16	R/W	See section	47.7.115/ 1974
A14	Message Buffer Frame ID Register (FR_MBFIDR66)	16	R/W	See section	47.7.116/ 1976
A16	Message Buffer Index Register (FR_MBIDXR66)	16	R/W	See section	47.7.117/ 1976
A18	Message Buffer Configuration, Control, Status Register (FR_MBCCSR67)	16	R/W	0000h	47.7.114/ 1972
A1A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR67)	16	R/W	See section	47.7.115/ 1974
A1C	Message Buffer Frame ID Register (FR_MBFIDR67)	16	R/W	See section	47.7.116/ 1976
A1E	Message Buffer Index Register (FR_MBIDXR67)	16	R/W	See section	47.7.117/ 1976
A20	Message Buffer Configuration, Control, Status Register (FR_MBCCSR68)	16	R/W	0000h	47.7.114/ 1972
A22	Message Buffer Cycle Counter Filter Register (FR_MBCCFR68)	16	R/W	See section	47.7.115/ 1974
A24	Message Buffer Frame ID Register (FR_MBFIDR68)	16	R/W	See section	47.7.116/ 1976
A26	Message Buffer Index Register (FR_MBIDXR68)	16	R/W	See section	47.7.117/ 1976
A28	Message Buffer Configuration, Control, Status Register (FR_MBCCSR69)	16	R/W	0000h	47.7.114/ 1972
A2A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR69)	16	R/W	See section	47.7.115/ 1974
A2C	Message Buffer Frame ID Register (FR_MBFIDR69)	16	R/W	See section	47.7.116/ 1976
A2E	Message Buffer Index Register (FR_MBIDXR69)	16	R/W	See section	47.7.117/ 1976
A30	Message Buffer Configuration, Control, Status Register (FR_MBCCSR70)	16	R/W	0000h	47.7.114/ 1972
A32	Message Buffer Cycle Counter Filter Register (FR_MBCCFR70)	16	R/W	See section	47.7.115/ 1974

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A34	Message Buffer Frame ID Register (FR_MBFIDR70)	16	R/W	See section	47.7.116/ 1976
A36	Message Buffer Index Register (FR_MBIDXR70)	16	R/W	See section	47.7.117/ 1976
A38	Message Buffer Configuration, Control, Status Register (FR_MBCCSR71)	16	R/W	0000h	47.7.114/ 1972
A3A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR71)	16	R/W	See section	47.7.115/ 1974
A3C	Message Buffer Frame ID Register (FR_MBFIDR71)	16	R/W	See section	47.7.116/ 1976
A3E	Message Buffer Index Register (FR_MBIDXR71)	16	R/W	See section	47.7.117/ 1976
A40	Message Buffer Configuration, Control, Status Register (FR_MBCCSR72)	16	R/W	0000h	47.7.114/ 1972
A42	Message Buffer Cycle Counter Filter Register (FR_MBCCFR72)	16	R/W	See section	47.7.115/ 1974
A44	Message Buffer Frame ID Register (FR_MBFIDR72)	16	R/W	See section	47.7.116/ 1976
A46	Message Buffer Index Register (FR_MBIDXR72)	16	R/W	See section	47.7.117/ 1976
A48	Message Buffer Configuration, Control, Status Register (FR_MBCCSR73)	16	R/W	0000h	47.7.114/ 1972
A4A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR73)	16	R/W	See section	47.7.115/ 1974
A4C	Message Buffer Frame ID Register (FR_MBFIDR73)	16	R/W	See section	47.7.116/ 1976
A4E	Message Buffer Index Register (FR_MBIDXR73)	16	R/W	See section	47.7.117/ 1976
A50	Message Buffer Configuration, Control, Status Register (FR_MBCCSR74)	16	R/W	0000h	47.7.114/ 1972
A52	Message Buffer Cycle Counter Filter Register (FR_MBCCFR74)	16	R/W	See section	47.7.115/ 1974
A54	Message Buffer Frame ID Register (FR_MBFIDR74)	16	R/W	See section	47.7.116/ 1976
A56	Message Buffer Index Register (FR_MBIDXR74)	16	R/W	See section	47.7.117/ 1976
A58	Message Buffer Configuration, Control, Status Register (FR_MBCCSR75)	16	R/W	0000h	47.7.114/ 1972
A5A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR75)	16	R/W	See section	47.7.115/ 1974
A5C	Message Buffer Frame ID Register (FR_MBFIDR75)	16	R/W	See section	47.7.116/ 1976
A5E	Message Buffer Index Register (FR_MBIDXR75)	16	R/W	See section	47.7.117/ 1976

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A60	Message Buffer Configuration, Control, Status Register (FR_MBCCSR76)	16	R/W	0000h	47.7.114/ 1972
A62	Message Buffer Cycle Counter Filter Register (FR_MBCCFR76)	16	R/W	See section	47.7.115/ 1974
A64	Message Buffer Frame ID Register (FR_MBFIDR76)	16	R/W	See section	47.7.116/ 1976
A66	Message Buffer Index Register (FR_MBIDXR76)	16	R/W	See section	47.7.117/ 1976
A68	Message Buffer Configuration, Control, Status Register (FR_MBCCSR77)	16	R/W	0000h	47.7.114/ 1972
A6A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR77)	16	R/W	See section	47.7.115/ 1974
A6C	Message Buffer Frame ID Register (FR_MBFIDR77)	16	R/W	See section	47.7.116/ 1976
A6E	Message Buffer Index Register (FR_MBIDXR77)	16	R/W	See section	47.7.117/ 1976
A70	Message Buffer Configuration, Control, Status Register (FR_MBCCSR78)	16	R/W	0000h	47.7.114/ 1972
A72	Message Buffer Cycle Counter Filter Register (FR_MBCCFR78)	16	R/W	See section	47.7.115/ 1974
A74	Message Buffer Frame ID Register (FR_MBFIDR78)	16	R/W	See section	47.7.116/ 1976
A76	Message Buffer Index Register (FR_MBIDXR78)	16	R/W	See section	47.7.117/ 1976
A78	Message Buffer Configuration, Control, Status Register (FR_MBCCSR79)	16	R/W	0000h	47.7.114/ 1972
A7A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR79)	16	R/W	See section	47.7.115/ 1974
A7C	Message Buffer Frame ID Register (FR_MBFIDR79)	16	R/W	See section	47.7.116/ 1976
A7E	Message Buffer Index Register (FR_MBIDXR79)	16	R/W	See section	47.7.117/ 1976
A80	Message Buffer Configuration, Control, Status Register (FR_MBCCSR80)	16	R/W	0000h	47.7.114/ 1972
A82	Message Buffer Cycle Counter Filter Register (FR_MBCCFR80)	16	R/W	See section	47.7.115/ 1974
A84	Message Buffer Frame ID Register (FR_MBFIDR80)	16	R/W	See section	47.7.116/ 1976
A86	Message Buffer Index Register (FR_MBIDXR80)	16	R/W	See section	47.7.117/ 1976
A88	Message Buffer Configuration, Control, Status Register (FR_MBCCSR81)	16	R/W	0000h	47.7.114/ 1972
A8A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR81)	16	R/W	See section	47.7.115/ 1974

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A8C	Message Buffer Frame ID Register (FR_MBFIDR81)	16	R/W	See section	47.7.116/ 1976
A8E	Message Buffer Index Register (FR_MBIDXR81)	16	R/W	See section	47.7.117/ 1976
A90	Message Buffer Configuration, Control, Status Register (FR_MBCCSR82)	16	R/W	0000h	47.7.114/ 1972
A92	Message Buffer Cycle Counter Filter Register (FR_MBCCFR82)	16	R/W	See section	47.7.115/ 1974
A94	Message Buffer Frame ID Register (FR_MBFIDR82)	16	R/W	See section	47.7.116/ 1976
A96	Message Buffer Index Register (FR_MBIDXR82)	16	R/W	See section	47.7.117/ 1976
A98	Message Buffer Configuration, Control, Status Register (FR_MBCCSR83)	16	R/W	0000h	47.7.114/ 1972
A9A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR83)	16	R/W	See section	47.7.115/ 1974
A9C	Message Buffer Frame ID Register (FR_MBFIDR83)	16	R/W	See section	47.7.116/ 1976
A9E	Message Buffer Index Register (FR_MBIDXR83)	16	R/W	See section	47.7.117/ 1976
AA0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR84)	16	R/W	0000h	47.7.114/ 1972
AA2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR84)	16	R/W	See section	47.7.115/ 1974
AA4	Message Buffer Frame ID Register (FR_MBFIDR84)	16	R/W	See section	47.7.116/ 1976
AA6	Message Buffer Index Register (FR_MBIDXR84)	16	R/W	See section	47.7.117/ 1976
AA8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR85)	16	R/W	0000h	47.7.114/ 1972
AAA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR85)	16	R/W	See section	47.7.115/ 1974
AAC	Message Buffer Frame ID Register (FR_MBFIDR85)	16	R/W	See section	47.7.116/ 1976
AAE	Message Buffer Index Register (FR_MBIDXR85)	16	R/W	See section	47.7.117/ 1976
AB0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR86)	16	R/W	0000h	47.7.114/ 1972
AB2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR86)	16	R/W	See section	47.7.115/ 1974
AB4	Message Buffer Frame ID Register (FR_MBFIDR86)	16	R/W	See section	47.7.116/ 1976
AB6	Message Buffer Index Register (FR_MBIDXR86)	16	R/W	See section	47.7.117/ 1976

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AB8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR87)	16	R/W	0000h	47.7.114/ 1972
ABA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR87)	16	R/W	See section	47.7.115/ 1974
ABC	Message Buffer Frame ID Register (FR_MBFIDR87)	16	R/W	See section	47.7.116/ 1976
ABE	Message Buffer Index Register (FR_MBIDXR87)	16	R/W	See section	47.7.117/ 1976
AC0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR88)	16	R/W	0000h	47.7.114/ 1972
AC2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR88)	16	R/W	See section	47.7.115/ 1974
AC4	Message Buffer Frame ID Register (FR_MBFIDR88)	16	R/W	See section	47.7.116/ 1976
AC6	Message Buffer Index Register (FR_MBIDXR88)	16	R/W	See section	47.7.117/ 1976
AC8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR89)	16	R/W	0000h	47.7.114/ 1972
ACA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR89)	16	R/W	See section	47.7.115/ 1974
ACC	Message Buffer Frame ID Register (FR_MBFIDR89)	16	R/W	See section	47.7.116/ 1976
ACE	Message Buffer Index Register (FR_MBIDXR89)	16	R/W	See section	47.7.117/ 1976
AD0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR90)	16	R/W	0000h	47.7.114/ 1972
AD2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR90)	16	R/W	See section	47.7.115/ 1974
AD4	Message Buffer Frame ID Register (FR_MBFIDR90)	16	R/W	See section	47.7.116/ 1976
AD6	Message Buffer Index Register (FR_MBIDXR90)	16	R/W	See section	47.7.117/ 1976
AD8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR91)	16	R/W	0000h	47.7.114/ 1972
ADA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR91)	16	R/W	See section	47.7.115/ 1974
ADC	Message Buffer Frame ID Register (FR_MBFIDR91)	16	R/W	See section	47.7.116/ 1976
ADE	Message Buffer Index Register (FR_MBIDXR91)	16	R/W	See section	47.7.117/ 1976
AE0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR92)	16	R/W	0000h	47.7.114/ 1972
AE2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR92)	16	R/W	See section	47.7.115/ 1974

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
AE4	Message Buffer Frame ID Register (FR_MBFIDR92)	16	R/W	See section	47.7.116/ 1976
AE6	Message Buffer Index Register (FR_MBIDXR92)	16	R/W	See section	47.7.117/ 1976
AE8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR93)	16	R/W	0000h	47.7.114/ 1972
AEA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR93)	16	R/W	See section	47.7.115/ 1974
AEC	Message Buffer Frame ID Register (FR_MBFIDR93)	16	R/W	See section	47.7.116/ 1976
AEE	Message Buffer Index Register (FR_MBIDXR93)	16	R/W	See section	47.7.117/ 1976
AF0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR94)	16	R/W	0000h	47.7.114/ 1972
AF2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR94)	16	R/W	See section	47.7.115/ 1974
AF4	Message Buffer Frame ID Register (FR_MBFIDR94)	16	R/W	See section	47.7.116/ 1976
AF6	Message Buffer Index Register (FR_MBIDXR94)	16	R/W	See section	47.7.117/ 1976
AF8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR95)	16	R/W	0000h	47.7.114/ 1972
AFA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR95)	16	R/W	See section	47.7.115/ 1974
AFC	Message Buffer Frame ID Register (FR_MBFIDR95)	16	R/W	See section	47.7.116/ 1976
AFE	Message Buffer Index Register (FR_MBIDXR95)	16	R/W	See section	47.7.117/ 1976
B00	Message Buffer Configuration, Control, Status Register (FR_MBCCSR96)	16	R/W	0000h	47.7.114/ 1972
B02	Message Buffer Cycle Counter Filter Register (FR_MBCCFR96)	16	R/W	See section	47.7.115/ 1974
B04	Message Buffer Frame ID Register (FR_MBFIDR96)	16	R/W	See section	47.7.116/ 1976
B06	Message Buffer Index Register (FR_MBIDXR96)	16	R/W	See section	47.7.117/ 1976
B08	Message Buffer Configuration, Control, Status Register (FR_MBCCSR97)	16	R/W	0000h	47.7.114/ 1972
B0A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR97)	16	R/W	See section	47.7.115/ 1974
B0C	Message Buffer Frame ID Register (FR_MBFIDR97)	16	R/W	See section	47.7.116/ 1976
B0E	Message Buffer Index Register (FR_MBIDXR97)	16	R/W	See section	47.7.117/ 1976

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B10	Message Buffer Configuration, Control, Status Register (FR_MBCCSR98)	16	R/W	0000h	47.7.114/ 1972
B12	Message Buffer Cycle Counter Filter Register (FR_MBCCFR98)	16	R/W	See section	47.7.115/ 1974
B14	Message Buffer Frame ID Register (FR_MBFIDR98)	16	R/W	See section	47.7.116/ 1976
B16	Message Buffer Index Register (FR_MBIDXR98)	16	R/W	See section	47.7.117/ 1976
B18	Message Buffer Configuration, Control, Status Register (FR_MBCCSR99)	16	R/W	0000h	47.7.114/ 1972
B1A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR99)	16	R/W	See section	47.7.115/ 1974
B1C	Message Buffer Frame ID Register (FR_MBFIDR99)	16	R/W	See section	47.7.116/ 1976
B1E	Message Buffer Index Register (FR_MBIDXR99)	16	R/W	See section	47.7.117/ 1976
B20	Message Buffer Configuration, Control, Status Register (FR_MBCCSR100)	16	R/W	0000h	47.7.114/ 1972
B22	Message Buffer Cycle Counter Filter Register (FR_MBCCFR100)	16	R/W	See section	47.7.115/ 1974
B24	Message Buffer Frame ID Register (FR_MBFIDR100)	16	R/W	See section	47.7.116/ 1976
B26	Message Buffer Index Register (FR_MBIDXR100)	16	R/W	See section	47.7.117/ 1976
B28	Message Buffer Configuration, Control, Status Register (FR_MBCCSR101)	16	R/W	0000h	47.7.114/ 1972
B2A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR101)	16	R/W	See section	47.7.115/ 1974
B2C	Message Buffer Frame ID Register (FR_MBFIDR101)	16	R/W	See section	47.7.116/ 1976
B2E	Message Buffer Index Register (FR_MBIDXR101)	16	R/W	See section	47.7.117/ 1976
B30	Message Buffer Configuration, Control, Status Register (FR_MBCCSR102)	16	R/W	0000h	47.7.114/ 1972
B32	Message Buffer Cycle Counter Filter Register (FR_MBCCFR102)	16	R/W	See section	47.7.115/ 1974
B34	Message Buffer Frame ID Register (FR_MBFIDR102)	16	R/W	See section	47.7.116/ 1976
B36	Message Buffer Index Register (FR_MBIDXR102)	16	R/W	See section	47.7.117/ 1976
B38	Message Buffer Configuration, Control, Status Register (FR_MBCCSR103)	16	R/W	0000h	47.7.114/ 1972
B3A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR103)	16	R/W	See section	47.7.115/ 1974

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B3C	Message Buffer Frame ID Register (FR_MBFIDR103)	16	R/W	See section	47.7.116/ 1976
B3E	Message Buffer Index Register (FR_MBIDXR103)	16	R/W	See section	47.7.117/ 1976
B40	Message Buffer Configuration, Control, Status Register (FR_MBCCSR104)	16	R/W	0000h	47.7.114/ 1972
B42	Message Buffer Cycle Counter Filter Register (FR_MBCCFR104)	16	R/W	See section	47.7.115/ 1974
B44	Message Buffer Frame ID Register (FR_MBFIDR104)	16	R/W	See section	47.7.116/ 1976
B46	Message Buffer Index Register (FR_MBIDXR104)	16	R/W	See section	47.7.117/ 1976
B48	Message Buffer Configuration, Control, Status Register (FR_MBCCSR105)	16	R/W	0000h	47.7.114/ 1972
B4A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR105)	16	R/W	See section	47.7.115/ 1974
B4C	Message Buffer Frame ID Register (FR_MBFIDR105)	16	R/W	See section	47.7.116/ 1976
B4E	Message Buffer Index Register (FR_MBIDXR105)	16	R/W	See section	47.7.117/ 1976
B50	Message Buffer Configuration, Control, Status Register (FR_MBCCSR106)	16	R/W	0000h	47.7.114/ 1972
B52	Message Buffer Cycle Counter Filter Register (FR_MBCCFR106)	16	R/W	See section	47.7.115/ 1974
B54	Message Buffer Frame ID Register (FR_MBFIDR106)	16	R/W	See section	47.7.116/ 1976
B56	Message Buffer Index Register (FR_MBIDXR106)	16	R/W	See section	47.7.117/ 1976
B58	Message Buffer Configuration, Control, Status Register (FR_MBCCSR107)	16	R/W	0000h	47.7.114/ 1972
B5A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR107)	16	R/W	See section	47.7.115/ 1974
B5C	Message Buffer Frame ID Register (FR_MBFIDR107)	16	R/W	See section	47.7.116/ 1976
B5E	Message Buffer Index Register (FR_MBIDXR107)	16	R/W	See section	47.7.117/ 1976
B60	Message Buffer Configuration, Control, Status Register (FR_MBCCSR108)	16	R/W	0000h	47.7.114/ 1972
B62	Message Buffer Cycle Counter Filter Register (FR_MBCCFR108)	16	R/W	See section	47.7.115/ 1974
B64	Message Buffer Frame ID Register (FR_MBFIDR108)	16	R/W	See section	47.7.116/ 1976
B66	Message Buffer Index Register (FR_MBIDXR108)	16	R/W	See section	47.7.117/ 1976

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B68	Message Buffer Configuration, Control, Status Register (FR_MBCCSR109)	16	R/W	0000h	47.7.114/ 1972
B6A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR109)	16	R/W	See section	47.7.115/ 1974
B6C	Message Buffer Frame ID Register (FR_MBFIDR109)	16	R/W	See section	47.7.116/ 1976
B6E	Message Buffer Index Register (FR_MBIDXR109)	16	R/W	See section	47.7.117/ 1976
B70	Message Buffer Configuration, Control, Status Register (FR_MBCCSR110)	16	R/W	0000h	47.7.114/ 1972
B72	Message Buffer Cycle Counter Filter Register (FR_MBCCFR110)	16	R/W	See section	47.7.115/ 1974
B74	Message Buffer Frame ID Register (FR_MBFIDR110)	16	R/W	See section	47.7.116/ 1976
B76	Message Buffer Index Register (FR_MBIDXR110)	16	R/W	See section	47.7.117/ 1976
B78	Message Buffer Configuration, Control, Status Register (FR_MBCCSR111)	16	R/W	0000h	47.7.114/ 1972
B7A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR111)	16	R/W	See section	47.7.115/ 1974
B7C	Message Buffer Frame ID Register (FR_MBFIDR111)	16	R/W	See section	47.7.116/ 1976
B7E	Message Buffer Index Register (FR_MBIDXR111)	16	R/W	See section	47.7.117/ 1976
B80	Message Buffer Configuration, Control, Status Register (FR_MBCCSR112)	16	R/W	0000h	47.7.114/ 1972
B82	Message Buffer Cycle Counter Filter Register (FR_MBCCFR112)	16	R/W	See section	47.7.115/ 1974
B84	Message Buffer Frame ID Register (FR_MBFIDR112)	16	R/W	See section	47.7.116/ 1976
B86	Message Buffer Index Register (FR_MBIDXR112)	16	R/W	See section	47.7.117/ 1976
B88	Message Buffer Configuration, Control, Status Register (FR_MBCCSR113)	16	R/W	0000h	47.7.114/ 1972
B8A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR113)	16	R/W	See section	47.7.115/ 1974
B8C	Message Buffer Frame ID Register (FR_MBFIDR113)	16	R/W	See section	47.7.116/ 1976
B8E	Message Buffer Index Register (FR_MBIDXR113)	16	R/W	See section	47.7.117/ 1976
B90	Message Buffer Configuration, Control, Status Register (FR_MBCCSR114)	16	R/W	0000h	47.7.114/ 1972
B92	Message Buffer Cycle Counter Filter Register (FR_MBCCFR114)	16	R/W	See section	47.7.115/ 1974

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
B94	Message Buffer Frame ID Register (FR_MBFIDR114)	16	R/W	See section	47.7.116/ 1976
B96	Message Buffer Index Register (FR_MBIDXR114)	16	R/W	See section	47.7.117/ 1976
B98	Message Buffer Configuration, Control, Status Register (FR_MBCCSR115)	16	R/W	0000h	47.7.114/ 1972
B9A	Message Buffer Cycle Counter Filter Register (FR_MBCCFR115)	16	R/W	See section	47.7.115/ 1974
B9C	Message Buffer Frame ID Register (FR_MBFIDR115)	16	R/W	See section	47.7.116/ 1976
B9E	Message Buffer Index Register (FR_MBIDXR115)	16	R/W	See section	47.7.117/ 1976
BA0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR116)	16	R/W	0000h	47.7.114/ 1972
BA2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR116)	16	R/W	See section	47.7.115/ 1974
BA4	Message Buffer Frame ID Register (FR_MBFIDR116)	16	R/W	See section	47.7.116/ 1976
BA6	Message Buffer Index Register (FR_MBIDXR116)	16	R/W	See section	47.7.117/ 1976
BA8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR117)	16	R/W	0000h	47.7.114/ 1972
BAA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR117)	16	R/W	See section	47.7.115/ 1974
BAC	Message Buffer Frame ID Register (FR_MBFIDR117)	16	R/W	See section	47.7.116/ 1976
BAE	Message Buffer Index Register (FR_MBIDXR117)	16	R/W	See section	47.7.117/ 1976
BB0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR118)	16	R/W	0000h	47.7.114/ 1972
BB2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR118)	16	R/W	See section	47.7.115/ 1974
BB4	Message Buffer Frame ID Register (FR_MBFIDR118)	16	R/W	See section	47.7.116/ 1976
BB6	Message Buffer Index Register (FR_MBIDXR118)	16	R/W	See section	47.7.117/ 1976
BB8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR119)	16	R/W	0000h	47.7.114/ 1972
BBA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR119)	16	R/W	See section	47.7.115/ 1974
BBC	Message Buffer Frame ID Register (FR_MBFIDR119)	16	R/W	See section	47.7.116/ 1976
BBE	Message Buffer Index Register (FR_MBIDXR119)	16	R/W	See section	47.7.117/ 1976

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
BC0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR120)	16	R/W	0000h	47.7.114/ 1972
BC2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR120)	16	R/W	See section	47.7.115/ 1974
BC4	Message Buffer Frame ID Register (FR_MBFIDR120)	16	R/W	See section	47.7.116/ 1976
BC6	Message Buffer Index Register (FR_MBIDXR120)	16	R/W	See section	47.7.117/ 1976
BC8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR121)	16	R/W	0000h	47.7.114/ 1972
BCA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR121)	16	R/W	See section	47.7.115/ 1974
BCC	Message Buffer Frame ID Register (FR_MBFIDR121)	16	R/W	See section	47.7.116/ 1976
BCE	Message Buffer Index Register (FR_MBIDXR121)	16	R/W	See section	47.7.117/ 1976
BD0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR122)	16	R/W	0000h	47.7.114/ 1972
BD2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR122)	16	R/W	See section	47.7.115/ 1974
BD4	Message Buffer Frame ID Register (FR_MBFIDR122)	16	R/W	See section	47.7.116/ 1976
BD6	Message Buffer Index Register (FR_MBIDXR122)	16	R/W	See section	47.7.117/ 1976
BD8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR123)	16	R/W	0000h	47.7.114/ 1972
BDA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR123)	16	R/W	See section	47.7.115/ 1974
BDC	Message Buffer Frame ID Register (FR_MBFIDR123)	16	R/W	See section	47.7.116/ 1976
BDE	Message Buffer Index Register (FR_MBIDXR123)	16	R/W	See section	47.7.117/ 1976
BE0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR124)	16	R/W	0000h	47.7.114/ 1972
BE2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR124)	16	R/W	See section	47.7.115/ 1974
BE4	Message Buffer Frame ID Register (FR_MBFIDR124)	16	R/W	See section	47.7.116/ 1976
BE6	Message Buffer Index Register (FR_MBIDXR124)	16	R/W	See section	47.7.117/ 1976
BE8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR125)	16	R/W	0000h	47.7.114/ 1972
BEA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR125)	16	R/W	See section	47.7.115/ 1974

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
BEC	Message Buffer Frame ID Register (FR_MBFIDR125)	16	R/W	See section	47.7.116/ 1976
BEE	Message Buffer Index Register (FR_MBIDXR125)	16	R/W	See section	47.7.117/ 1976
BF0	Message Buffer Configuration, Control, Status Register (FR_MBCCSR126)	16	R/W	0000h	47.7.114/ 1972
BF2	Message Buffer Cycle Counter Filter Register (FR_MBCCFR126)	16	R/W	See section	47.7.115/ 1974
BF4	Message Buffer Frame ID Register (FR_MBFIDR126)	16	R/W	See section	47.7.116/ 1976
BF6	Message Buffer Index Register (FR_MBIDXR126)	16	R/W	See section	47.7.117/ 1976
BF8	Message Buffer Configuration, Control, Status Register (FR_MBCCSR127)	16	R/W	0000h	47.7.114/ 1972
BFA	Message Buffer Cycle Counter Filter Register (FR_MBCCFR127)	16	R/W	See section	47.7.115/ 1974
BFC	Message Buffer Frame ID Register (FR_MBFIDR127)	16	R/W	See section	47.7.116/ 1976
BFE	Message Buffer Index Register (FR_MBIDXR127)	16	R/W	See section	47.7.117/ 1976
1000	Message Buffer Data Field Offset Register (FR_MBDOR0)	16	R/W	See section	47.7.118/ 1977
1002	Message Buffer Data Field Offset Register (FR_MBDOR1)	16	R/W	See section	47.7.118/ 1977
1004	Message Buffer Data Field Offset Register (FR_MBDOR2)	16	R/W	See section	47.7.118/ 1977
1006	Message Buffer Data Field Offset Register (FR_MBDOR3)	16	R/W	See section	47.7.118/ 1977
1008	Message Buffer Data Field Offset Register (FR_MBDOR4)	16	R/W	See section	47.7.118/ 1977
100A	Message Buffer Data Field Offset Register (FR_MBDOR5)	16	R/W	See section	47.7.118/ 1977
100C	Message Buffer Data Field Offset Register (FR_MBDOR6)	16	R/W	See section	47.7.118/ 1977
100E	Message Buffer Data Field Offset Register (FR_MBDOR7)	16	R/W	See section	47.7.118/ 1977
1010	Message Buffer Data Field Offset Register (FR_MBDOR8)	16	R/W	See section	47.7.118/ 1977
1012	Message Buffer Data Field Offset Register (FR_MBDOR9)	16	R/W	See section	47.7.118/ 1977
1014	Message Buffer Data Field Offset Register (FR_MBDOR10)	16	R/W	See section	47.7.118/ 1977
1016	Message Buffer Data Field Offset Register (FR_MBDOR11)	16	R/W	See section	47.7.118/ 1977

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1018	Message Buffer Data Field Offset Register (FR_MBDOR12)	16	R/W	See section	47.7.118/ 1977
101A	Message Buffer Data Field Offset Register (FR_MBDOR13)	16	R/W	See section	47.7.118/ 1977
101C	Message Buffer Data Field Offset Register (FR_MBDOR14)	16	R/W	See section	47.7.118/ 1977
101E	Message Buffer Data Field Offset Register (FR_MBDOR15)	16	R/W	See section	47.7.118/ 1977
1020	Message Buffer Data Field Offset Register (FR_MBDOR16)	16	R/W	See section	47.7.118/ 1977
1022	Message Buffer Data Field Offset Register (FR_MBDOR17)	16	R/W	See section	47.7.118/ 1977
1024	Message Buffer Data Field Offset Register (FR_MBDOR18)	16	R/W	See section	47.7.118/ 1977
1026	Message Buffer Data Field Offset Register (FR_MBDOR19)	16	R/W	See section	47.7.118/ 1977
1028	Message Buffer Data Field Offset Register (FR_MBDOR20)	16	R/W	See section	47.7.118/ 1977
102A	Message Buffer Data Field Offset Register (FR_MBDOR21)	16	R/W	See section	47.7.118/ 1977
102C	Message Buffer Data Field Offset Register (FR_MBDOR22)	16	R/W	See section	47.7.118/ 1977
102E	Message Buffer Data Field Offset Register (FR_MBDOR23)	16	R/W	See section	47.7.118/ 1977
1030	Message Buffer Data Field Offset Register (FR_MBDOR24)	16	R/W	See section	47.7.118/ 1977
1032	Message Buffer Data Field Offset Register (FR_MBDOR25)	16	R/W	See section	47.7.118/ 1977
1034	Message Buffer Data Field Offset Register (FR_MBDOR26)	16	R/W	See section	47.7.118/ 1977
1036	Message Buffer Data Field Offset Register (FR_MBDOR27)	16	R/W	See section	47.7.118/ 1977
1038	Message Buffer Data Field Offset Register (FR_MBDOR28)	16	R/W	See section	47.7.118/ 1977
103A	Message Buffer Data Field Offset Register (FR_MBDOR29)	16	R/W	See section	47.7.118/ 1977
103C	Message Buffer Data Field Offset Register (FR_MBDOR30)	16	R/W	See section	47.7.118/ 1977
103E	Message Buffer Data Field Offset Register (FR_MBDOR31)	16	R/W	See section	47.7.118/ 1977
1040	Message Buffer Data Field Offset Register (FR_MBDOR32)	16	R/W	See section	47.7.118/ 1977
1042	Message Buffer Data Field Offset Register (FR_MBDOR33)	16	R/W	See section	47.7.118/ 1977

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1044	Message Buffer Data Field Offset Register (FR_MBDOR34)	16	R/W	See section	47.7.118/ 1977
1046	Message Buffer Data Field Offset Register (FR_MBDOR35)	16	R/W	See section	47.7.118/ 1977
1048	Message Buffer Data Field Offset Register (FR_MBDOR36)	16	R/W	See section	47.7.118/ 1977
104A	Message Buffer Data Field Offset Register (FR_MBDOR37)	16	R/W	See section	47.7.118/ 1977
104C	Message Buffer Data Field Offset Register (FR_MBDOR38)	16	R/W	See section	47.7.118/ 1977
104E	Message Buffer Data Field Offset Register (FR_MBDOR39)	16	R/W	See section	47.7.118/ 1977
1050	Message Buffer Data Field Offset Register (FR_MBDOR40)	16	R/W	See section	47.7.118/ 1977
1052	Message Buffer Data Field Offset Register (FR_MBDOR41)	16	R/W	See section	47.7.118/ 1977
1054	Message Buffer Data Field Offset Register (FR_MBDOR42)	16	R/W	See section	47.7.118/ 1977
1056	Message Buffer Data Field Offset Register (FR_MBDOR43)	16	R/W	See section	47.7.118/ 1977
1058	Message Buffer Data Field Offset Register (FR_MBDOR44)	16	R/W	See section	47.7.118/ 1977
105A	Message Buffer Data Field Offset Register (FR_MBDOR45)	16	R/W	See section	47.7.118/ 1977
105C	Message Buffer Data Field Offset Register (FR_MBDOR46)	16	R/W	See section	47.7.118/ 1977
105E	Message Buffer Data Field Offset Register (FR_MBDOR47)	16	R/W	See section	47.7.118/ 1977
1060	Message Buffer Data Field Offset Register (FR_MBDOR48)	16	R/W	See section	47.7.118/ 1977
1062	Message Buffer Data Field Offset Register (FR_MBDOR49)	16	R/W	See section	47.7.118/ 1977
1064	Message Buffer Data Field Offset Register (FR_MBDOR50)	16	R/W	See section	47.7.118/ 1977
1066	Message Buffer Data Field Offset Register (FR_MBDOR51)	16	R/W	See section	47.7.118/ 1977
1068	Message Buffer Data Field Offset Register (FR_MBDOR52)	16	R/W	See section	47.7.118/ 1977
106A	Message Buffer Data Field Offset Register (FR_MBDOR53)	16	R/W	See section	47.7.118/ 1977
106C	Message Buffer Data Field Offset Register (FR_MBDOR54)	16	R/W	See section	47.7.118/ 1977
106E	Message Buffer Data Field Offset Register (FR_MBDOR55)	16	R/W	See section	47.7.118/ 1977

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1070	Message Buffer Data Field Offset Register (FR_MBDOR56)	16	R/W	See section	47.7.118/ 1977
1072	Message Buffer Data Field Offset Register (FR_MBDOR57)	16	R/W	See section	47.7.118/ 1977
1074	Message Buffer Data Field Offset Register (FR_MBDOR58)	16	R/W	See section	47.7.118/ 1977
1076	Message Buffer Data Field Offset Register (FR_MBDOR59)	16	R/W	See section	47.7.118/ 1977
1078	Message Buffer Data Field Offset Register (FR_MBDOR60)	16	R/W	See section	47.7.118/ 1977
107A	Message Buffer Data Field Offset Register (FR_MBDOR61)	16	R/W	See section	47.7.118/ 1977
107C	Message Buffer Data Field Offset Register (FR_MBDOR62)	16	R/W	See section	47.7.118/ 1977
107E	Message Buffer Data Field Offset Register (FR_MBDOR63)	16	R/W	See section	47.7.118/ 1977
1080	Message Buffer Data Field Offset Register (FR_MBDOR64)	16	R/W	See section	47.7.118/ 1977
1082	Message Buffer Data Field Offset Register (FR_MBDOR65)	16	R/W	See section	47.7.118/ 1977
1084	Message Buffer Data Field Offset Register (FR_MBDOR66)	16	R/W	See section	47.7.118/ 1977
1086	Message Buffer Data Field Offset Register (FR_MBDOR67)	16	R/W	See section	47.7.118/ 1977
1088	Message Buffer Data Field Offset Register (FR_MBDOR68)	16	R/W	See section	47.7.118/ 1977
108A	Message Buffer Data Field Offset Register (FR_MBDOR69)	16	R/W	See section	47.7.118/ 1977
108C	Message Buffer Data Field Offset Register (FR_MBDOR70)	16	R/W	See section	47.7.118/ 1977
108E	Message Buffer Data Field Offset Register (FR_MBDOR71)	16	R/W	See section	47.7.118/ 1977
1090	Message Buffer Data Field Offset Register (FR_MBDOR72)	16	R/W	See section	47.7.118/ 1977
1092	Message Buffer Data Field Offset Register (FR_MBDOR73)	16	R/W	See section	47.7.118/ 1977
1094	Message Buffer Data Field Offset Register (FR_MBDOR74)	16	R/W	See section	47.7.118/ 1977
1096	Message Buffer Data Field Offset Register (FR_MBDOR75)	16	R/W	See section	47.7.118/ 1977
1098	Message Buffer Data Field Offset Register (FR_MBDOR76)	16	R/W	See section	47.7.118/ 1977
109A	Message Buffer Data Field Offset Register (FR_MBDOR77)	16	R/W	See section	47.7.118/ 1977

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
109C	Message Buffer Data Field Offset Register (FR_MBDOR78)	16	R/W	See section	47.7.118/ 1977
109E	Message Buffer Data Field Offset Register (FR_MBDOR79)	16	R/W	See section	47.7.118/ 1977
10A0	Message Buffer Data Field Offset Register (FR_MBDOR80)	16	R/W	See section	47.7.118/ 1977
10A2	Message Buffer Data Field Offset Register (FR_MBDOR81)	16	R/W	See section	47.7.118/ 1977
10A4	Message Buffer Data Field Offset Register (FR_MBDOR82)	16	R/W	See section	47.7.118/ 1977
10A6	Message Buffer Data Field Offset Register (FR_MBDOR83)	16	R/W	See section	47.7.118/ 1977
10A8	Message Buffer Data Field Offset Register (FR_MBDOR84)	16	R/W	See section	47.7.118/ 1977
10AA	Message Buffer Data Field Offset Register (FR_MBDOR85)	16	R/W	See section	47.7.118/ 1977
10AC	Message Buffer Data Field Offset Register (FR_MBDOR86)	16	R/W	See section	47.7.118/ 1977
10AE	Message Buffer Data Field Offset Register (FR_MBDOR87)	16	R/W	See section	47.7.118/ 1977
10B0	Message Buffer Data Field Offset Register (FR_MBDOR88)	16	R/W	See section	47.7.118/ 1977
10B2	Message Buffer Data Field Offset Register (FR_MBDOR89)	16	R/W	See section	47.7.118/ 1977
10B4	Message Buffer Data Field Offset Register (FR_MBDOR90)	16	R/W	See section	47.7.118/ 1977
10B6	Message Buffer Data Field Offset Register (FR_MBDOR91)	16	R/W	See section	47.7.118/ 1977
10B8	Message Buffer Data Field Offset Register (FR_MBDOR92)	16	R/W	See section	47.7.118/ 1977
10BA	Message Buffer Data Field Offset Register (FR_MBDOR93)	16	R/W	See section	47.7.118/ 1977
10BC	Message Buffer Data Field Offset Register (FR_MBDOR94)	16	R/W	See section	47.7.118/ 1977
10BE	Message Buffer Data Field Offset Register (FR_MBDOR95)	16	R/W	See section	47.7.118/ 1977
10C0	Message Buffer Data Field Offset Register (FR_MBDOR96)	16	R/W	See section	47.7.118/ 1977
10C2	Message Buffer Data Field Offset Register (FR_MBDOR97)	16	R/W	See section	47.7.118/ 1977
10C4	Message Buffer Data Field Offset Register (FR_MBDOR98)	16	R/W	See section	47.7.118/ 1977
10C6	Message Buffer Data Field Offset Register (FR_MBDOR99)	16	R/W	See section	47.7.118/ 1977

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10C8	Message Buffer Data Field Offset Register (FR_MBDOR100)	16	R/W	See section	47.7.118/ 1977
10CA	Message Buffer Data Field Offset Register (FR_MBDOR101)	16	R/W	See section	47.7.118/ 1977
10CC	Message Buffer Data Field Offset Register (FR_MBDOR102)	16	R/W	See section	47.7.118/ 1977
10CE	Message Buffer Data Field Offset Register (FR_MBDOR103)	16	R/W	See section	47.7.118/ 1977
10D0	Message Buffer Data Field Offset Register (FR_MBDOR104)	16	R/W	See section	47.7.118/ 1977
10D2	Message Buffer Data Field Offset Register (FR_MBDOR105)	16	R/W	See section	47.7.118/ 1977
10D4	Message Buffer Data Field Offset Register (FR_MBDOR106)	16	R/W	See section	47.7.118/ 1977
10D6	Message Buffer Data Field Offset Register (FR_MBDOR107)	16	R/W	See section	47.7.118/ 1977
10D8	Message Buffer Data Field Offset Register (FR_MBDOR108)	16	R/W	See section	47.7.118/ 1977
10DA	Message Buffer Data Field Offset Register (FR_MBDOR109)	16	R/W	See section	47.7.118/ 1977
10DC	Message Buffer Data Field Offset Register (FR_MBDOR110)	16	R/W	See section	47.7.118/ 1977
10DE	Message Buffer Data Field Offset Register (FR_MBDOR111)	16	R/W	See section	47.7.118/ 1977
10E0	Message Buffer Data Field Offset Register (FR_MBDOR112)	16	R/W	See section	47.7.118/ 1977
10E2	Message Buffer Data Field Offset Register (FR_MBDOR113)	16	R/W	See section	47.7.118/ 1977
10E4	Message Buffer Data Field Offset Register (FR_MBDOR114)	16	R/W	See section	47.7.118/ 1977
10E6	Message Buffer Data Field Offset Register (FR_MBDOR115)	16	R/W	See section	47.7.118/ 1977
10E8	Message Buffer Data Field Offset Register (FR_MBDOR116)	16	R/W	See section	47.7.118/ 1977
10EA	Message Buffer Data Field Offset Register (FR_MBDOR117)	16	R/W	See section	47.7.118/ 1977
10EC	Message Buffer Data Field Offset Register (FR_MBDOR118)	16	R/W	See section	47.7.118/ 1977
10EE	Message Buffer Data Field Offset Register (FR_MBDOR119)	16	R/W	See section	47.7.118/ 1977
10F0	Message Buffer Data Field Offset Register (FR_MBDOR120)	16	R/W	See section	47.7.118/ 1977
10F2	Message Buffer Data Field Offset Register (FR_MBDOR121)	16	R/W	See section	47.7.118/ 1977

Table continues on the next page...

FR memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10F4	Message Buffer Data Field Offset Register (FR_MBDOR122)	16	R/W	See section	47.7.118/ 1977
10F6	Message Buffer Data Field Offset Register (FR_MBDOR123)	16	R/W	See section	47.7.118/ 1977
10F8	Message Buffer Data Field Offset Register (FR_MBDOR124)	16	R/W	See section	47.7.118/ 1977
10FA	Message Buffer Data Field Offset Register (FR_MBDOR125)	16	R/W	See section	47.7.118/ 1977
10FC	Message Buffer Data Field Offset Register (FR_MBDOR126)	16	R/W	See section	47.7.118/ 1977
10FE	Message Buffer Data Field Offset Register (FR_MBDOR127)	16	R/W	See section	47.7.118/ 1977
1100	Message Buffer Data Field Offset Register (FR_MBDOR128)	16	R/W	See section	47.7.118/ 1977
1102	Message Buffer Data Field Offset Register (FR_MBDOR129)	16	R/W	See section	47.7.118/ 1977
1104	Message Buffer Data Field Offset Register (FR_MBDOR130)	16	R/W	See section	47.7.118/ 1977
1106	Message Buffer Data Field Offset Register (FR_MBDOR131)	16	R/W	See section	47.7.118/ 1977
1108	LRAM ECC Error Test Register (FR_LEETR0)	16	R/W	See section	47.7.119/ 1978
110A	LRAM ECC Error Test Register (FR_LEETR1)	16	R/W	See section	47.7.119/ 1978
110C	LRAM ECC Error Test Register (FR_LEETR2)	16	R/W	See section	47.7.119/ 1978
110E	LRAM ECC Error Test Register (FR_LEETR3)	16	R/W	See section	47.7.119/ 1978
1110	LRAM ECC Error Test Register (FR_LEETR4)	16	R/W	See section	47.7.119/ 1978
1112	LRAM ECC Error Test Register (FR_LEETR5)	16	R/W	See section	47.7.119/ 1978

47.7.1 Module Version Register (FR_MVR)

This register provides the CC version number. The module version number is derived from the CHI version number and the PE version number.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CHIVER								PEVER							
Write																
Reset	1	0	1	0	0	1	0	1	0	1	1	0	1	0	0	0

FR_MVR field descriptions

Field	Description
0–7 CHIVER	CHI Version Number. This field provides the version number of the controller host interface.
8–15 PEVER	PE Version Number. This field provides the version number of the protocol engine.

47.7.2 Module Configuration Register (FR_MCR)

This register defines the global configuration of the CC.

Write: SBFF, SCM, CHB, CHA, ECCE, FUM, FAM , CLKSEL , BITRATE: Disabled Mode and SFFE, MEN: Disabled Mode or POC: config

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7
Read Write	MEN	SBFF	SCM	CHB	CHA	SFFE	ECCE	Reserved
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	FUM	FAM	0	CLKSEL	BITRATE			0
Write								
Reset	0	0	0	0	0	0	0	0

FR_MCR field descriptions

Field	Description
0 MEN	Module Enable. This bit indicates whether or not the CC is in the Disabled Mode. The application requests the CC to leave the Disabled Mode by writing 1 to this bit Before leaving the Disabled Mode, the

Table continues on the next page...

FR_MCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>application must configure the SCM, SBFF, CHB, CHA, TMODE, BITRATE values. For details see Modes of Operation.

NOTE: If the CC is enabled it can only be disabled during mode: POC:default config.

0 Write: only during POC:default config, CC disable Read: CC disabled
1 Write: enable CC Read: CC enabled</td></tr>
<tr><td>1
SBFF</td><td>System Bus Failure Freeze

This bit controls the behavior of the CC in case of a system bus failure.

0 Continue normal operation
1 Transition to freeze mode</td></tr>
<tr><td>2
SCM</td><td>Single Channel Device Mode. This control bit defines the channel device mode of the CC as described in Channel Device Modes.

0 CC works in dual channel device mode
1 CC works in single channel device mode</td></tr>
<tr><td>3
CHB</td><td>Channel B Enable. protocol related parameter: pChannels The semantic of these control bits depends on the channel device mode controlled by the SCM bit and is given in the following table.
Table 47-7. FlexRay channel selection
<table>
<tr><th>SCM</th><th>CHB</th><th>CHA</th><th>Description</th></tr>
<tr><td colspan="4">Dual Channel Device Modes</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC
ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC</td></tr>
<tr><td></td><td>0</td><td>1</td><td>ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC - connected to FlexRay channel A
ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC</td></tr>
<tr><td></td><td>1</td><td>0</td><td>ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC
ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC - connected to FlexRay channel B</td></tr>
<tr><td></td><td>1</td><td>1</td><td>ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC - connected to FlexRay channel A
ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC - connected to FlexRay channel B</td></tr>
<tr><td colspan="4">Single Channel Device Mode</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC
ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC</td></tr>
<tr><td></td><td>0</td><td>1</td><td>ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC - connected to FlexRay channel A
ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC</td></tr>
<tr><td></td><td>1</td><td>0</td><td>ports FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ driven by CC - connected to FlexRay channel B
ports FR_B_RX, FR_B_TX, and $\overline{\text{FR_A_TX_EN}}$ not driven by CC</td></tr>
<tr><td></td><td>1</td><td>1</td><td>reserved</td></tr>
</table></td></tr>
</table>

Table continues on the next page...

FR_MCR field descriptions (continued)

Field	Description
4 CHA	Channel A Enable. protocol related parameter: pChannels The semantic of these control bits depends on the channel device mode controlled by the SCM bit and is given in the FlexRay channel selection table as seen above.
5 SFFE	Synchronization Frame Filter Enable. This bit controls the filtering for received synchronization frames. For details see Sync Frame Filtering . 0 Synchronization frame filtering disabled 1 Synchronization frame filtering enabled
6 ECCE	ECC Functionality Enable. This bit controls the ECC memory error detection functionality. For details see Memory Content Fault Detection . 0 ECC functionality (injection, detection, reporting, response) disabled 1 ECC functionality enabled
7 Reserved	Reserved bit, will not be changed. Application must not write any value different from the reset value. This field is reserved.
8 FUM	FIFO Update Mode. This bit controls the FIFO update behavior when the interrupt flags FR_GIFER[FAFAIF] and FR_GIFER[FAFBIF] are written by the application (see FIFO Update) 0 FIFOA/FIFOB is updated on writing 1 to FR_GIFER[FAFAIF] /FR_GIFER[FAFBIF] 1 FIFOA/FIFOB) is not updated on writing 1 to FR_GIFER[FAFAIF]/FR_GIFER[FAFBIF]
9 FAM	FIFO Address Mode. This bit controls the location of the system memory base address for the FIFOs. (see FIFO Configuration) 0 FIFO Base Address located in System Memory Base Address High Register (FR_SYMBADHR)) 1 FIFO Base Address located in Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 CLKSEL	Protocol Engine Clock Source Select. This bit is used to select the clock source for the protocol engine. 0 PE clock source is generated by on-chip crystal oscillator. 1 PE clock source is generated by on-chip PLL.
12–14 BITRATE	FlexRay Bus Bit Rate. This bit field defines the FlexRay Bus Bit Rate. 000 10.0 Mbit/sec 001 5.0 Mbit/sec 010 2.5 Mbit/sec 011 8.0 Mbit/sec 100 reserved 101 reserved 110 reserved 111 reserved
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.7.3 System Memory Base Address High Register (FR_SYMBADHR)

NOTE

The system memory base address must be set before the CC is enabled.

The system memory base address registers (FR_SYMBADHR) define the base address of the FlexRay memory area within the system memory. The base address is used by the BMIF to calculate the physical memory address for system memory accesses.

The FR_SYMBADHR register contains the most significant bits of the base address.

Write: Disabled Mode

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SMBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SYMBADHR field descriptions

Field	Description
0–15 SMBA	System Memory Base Address high. This is the value of the system memory base address for the individual message buffers and sync frame table. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.

47.7.4 System Memory Base Address Low Register (FR_SYMBADLR)

NOTE

The system memory base address must be set before the CC is enabled.

The system memory base address registers (FR_SYMBADLR) define the base address of the FlexRay memory area within the system memory. The base address is used by the BMIF to calculate the physical memory address for system memory accesses.

The FR_SYMBADLR register contains the least significant bits of the base address.

Write: Disabled Mode

Address: 0h base + 6h offset = 6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SMBA												0			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SYMBADLR field descriptions

Field	Description
0–11 SMBA	System Memory Base Address low. This is the value of the system memory base address for the individual message buffers and sync frame table. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.7.5 Strobe Signal Control Register (FR_STBSCR)

Write: Anytime

This register is used to assign the individual protocol timing related strobe signals given in Table 47-8 to the external strobe ports. Each strobe signal can be assigned to at most one strobe port . Each write access to registers overwrites the previously written ENB and STBPSEL values for the signal indicated by SEL . If more than one strobe signal is assigned to one strobe port, the current values of the strobe signals are combined with a binary OR and presented at the strobe port . If no strobe signal is assigned to a strobe port, the strobe port carries logic 0 . For more detailed and timing information refer to Strobe Signal Support .

NOTE

In single channel device mode, channel B related strobe signals are undefined and should not be assigned to the strobe ports.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0			SEL				0			ENB	0		STBPSEL	
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_STBSCR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.

Table continues on the next page...

FR_STBSCR field descriptions (continued)

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>1–3
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>4–7
SEL</td><td>Strobe Signal Select. This control field selects one of the strobe signals given in Table 47-8 to be enabled or disabled and assigned to one of the four strobe ports given in the following table.
Table 47-8. Strobe Signal Mapping
<table>
<tr><th colspan="2">SEL</th><th rowspan="2">Description</th><th rowspan="2">Channel</th><th rowspan="2">Type</th><th rowspan="2">Offset (Given in PE clock cycles)</th><th rowspan="2">Reference</th></tr>
<tr><th>dec</th><th>hex</th></tr>
<tr><td>0</td><td>0x0</td><td>arm</td><td>-</td><td>value</td><td>+1</td><td>MT start</td></tr>
<tr><td>1</td><td>0x1</td><td>mt</td><td>-</td><td>value</td><td>+1</td><td>MT start</td></tr>
<tr><td>2</td><td>0x2</td><td>cycle start</td><td>-</td><td>pulse</td><td>0</td><td>MT start</td></tr>
<tr><td>3</td><td>0x3</td><td>minislot start</td><td>-</td><td>pulse</td><td>0</td><td>MT start</td></tr>
<tr><td>4</td><td>0x4</td><td rowspan="2">slot start</td><td>A</td><td rowspan="2">pulse</td><td rowspan="2">0</td><td rowspan="2">MT start</td></tr>
<tr><td>5</td><td>0x5</td><td>B</td></tr>
<tr><td>6</td><td>0x6</td><td rowspan="2">receive data after glitch filtering</td><td>A</td><td rowspan="2">value</td><td rowspan="2">+4</td><td>FR_A_RX</td></tr>
<tr><td>7</td><td>0x7</td><td>B</td><td>FR_B_RX</td></tr>
<tr><td>8</td><td>0x8</td><td rowspan="2">channel idle indicator</td><td>A</td><td rowspan="2">level</td><td rowspan="2">+5</td><td>FR_A_RX</td></tr>
<tr><td>9</td><td>0x9</td><td>B</td><td>FR_B_RX</td></tr>
<tr><td>10</td><td>0xA</td><td rowspan="2">syntax error detected</td><td>A</td><td rowspan="2">pulse</td><td rowspan="2">+4</td><td>FR_A_RX</td></tr>
<tr><td>11</td><td>0xB</td><td>B</td><td>FR_B_RX</td></tr>
<tr><td>12</td><td>0xC</td><td rowspan="2">content error detected</td><td>A</td><td rowspan="2">level</td><td rowspan="2">+4</td><td>FR_A_RX</td></tr>
<tr><td>13</td><td>0xD</td><td>B</td><td>FR_B_RX</td></tr>
<tr><td>14</td><td>0xE</td><td rowspan="2">receive FIFO almost-full interrupt signals</td><td>A</td><td rowspan="2">value</td><td rowspan="2">n.a.</td><td>RX FIFO A Almost Full Interrupt</td></tr>
<tr><td>15</td><td>0xF</td><td>B</td><td>RX FIFO B Almost Full Interrupt</td></tr>
</table>
Write: Disabled Mode</td></tr>
<tr><td>8–10
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
<tr><td>11
ENB</td><td>Strobe Signal Enable. The control bit is used to enable and to disable the strobe signal selected by STBSSEL.

0 Strobe signal is disabled and not assigned to any strobe port.
1 Strobe signal is enabled and assigned to the strobe port selected by STBPSEL.</td></tr>
<tr><td>12–13
Reserved</td><td>This field is reserved.
This read-only field is reserved and always has the value 0.</td></tr>
</table>

Table continues on the next page...

FR_STBSCR field descriptions (continued)

Field	Description
14–15 STBPSEL	Strobe Port Select. This field selects the strobe port that the strobe signal selected by the SEL is assigned to. All strobe signals that are enabled and assigned to the same strobe port are combined with a binary OR operation. 00 assign selected signal to FR_DBG[0] 01 assign selected signal to FR_DBG[1] 10 assign selected signal to FR_DBG[2] 11 assign selected signal to FR_DBG[3]

47.7.6 Message Buffer Data Size Register (FR_MBDSR)

Write: POC:config

This register defines the size of the message buffer data section for the two message buffer segments in a number of two-byte entities.

The CC provides two independent segments for the individual message buffers. All individual message buffers within one segment have to have the same size for the message buffer data section. This size can be different for the two message buffer segments.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	MBSEG2DS							0	MBSEG1DS						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MBDSR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 MBSEG2DS	Message Buffer Segment 2 Data Size. The field defines the size of the message buffer data section in two-byte entities for message buffers within the second message buffer segment.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 MBSEG1DS	Message Buffer Segment 1 Data Size. The field defines the size of the message buffer data section in two-byte entities for message buffers within the first message buffer segment.

47.7.7 Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)

Write: POC:config

This register is used to define the last individual message buffer that belongs to the first message buffer segment and the number of the last used individual message buffer.

Address: 0h base + Eh offset = Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	LAST_MB_SEG1							0	LAST_MB_UTIL						
Write																
Reset	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1

FR_MBSSUTR field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 LAST_MB_SEG1	Last Message Buffer In Segment 1. This field defines the message buffer number of the last individual message buffer that is assigned to the first message buffer segment. The individual message buffers in the first segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn with n <= LAST_MB_SEG1. The first message buffer segment contains LAST_MB_SEG1+1 individual message buffers. **NOTE:** The first message buffer segment contains at least one individual message buffer. The individual message buffers in the second message buffer segment correspond to the message buffer control registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn with LAST_MB_SEG1 < n < 128 . **NOTE:** If LAST_MB_SEG1 = 127 all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 LAST_MB_UTIL	Last Message Buffer Utilized. This field defines the message buffer number of last utilized individual message buffer. The message buffer search engine examines all individual message buffer with a message buffer number n <= LAST_MB_UTIL. **NOTE:** If LAST_MB_UTIL=LAST_MB_SEG1 all individual message buffers belong to the first message buffer segment and the second message buffer segment is empty.

47.7.8 PE DRAM Access Register (FR_PEDRAR)

The PEDRAR register is used to trigger write and read operations on the PE data memory (PE DRAM). These operations are used for memory error injection and memory error observation.

Each write access to this registers initiates a read or write operation on the PE DRAM. The access done status bit DAD is cleared after the write access and is set if the PE DRAM access has been finished.

In case of an PE DRAM write access, the data provided in FR_PEDRDR are written into the PE DRAM, read back from the PE DRAM and are stored into the FR_PEDRDR.

In case of an PE DRAM read access, the requested data are read from PE DRAM and stored into the FR_PEDRDR.

For a detailed description refer to Memory Content Fault Detection

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	INST				ADDR											DAD
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PEDRAR field descriptions

Field	Description
0–3 INST	PE DRAM Access Instruction. This field defines the operation to be executed on the PE DRAM. 0011 PE DRAM write: Write FR_PEDRDR[DATA] to PE DRAM address ADDR (16 bit) 0101 PE DRAM read: Read Data from PE DRAM address ADDR (16 bit) into FR_PEDRDR[DATA] other reserved
4–14 ADDR	PE DRAM Access Address. This field defines the address in the PE DRAM to be written to or read from.
15 DAD	PE DRAM Access Done. This status bit is cleared when the application has written to this register and is set when the PE DRAM access has finished. 0 PE DRAM access running 1 PE DRAM access done

47.7.9 PE DRAM Data Register (FR_PEDRDR)

This register provides the data to be written to or read from the PE DRAM by the access initiated by write access to the FR_PEDRAR.

Address: 0h base + 12h offset = 12h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PEDRDR field descriptions

Field	Description
0–15 DATA	Data to be written to or read from the PE DRAM by the access initiated by write access to the FR_PEDRAR.

47.7.10 Protocol Operation Control Register (FR_POCR)

The application uses this register to issue

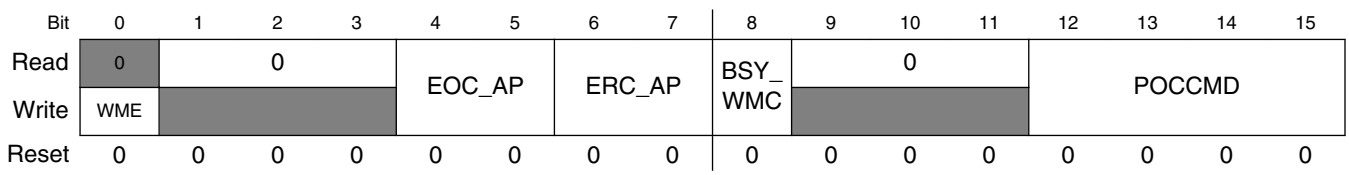
- protocol control commands
- external clock correction commands

Protocol control commands are issued by writing to the POCCMD field. For more information on protocol control commands, see Protocol Control Command Execution .

External clock correction commands are issued by writing to the EOC_AP and ERC_AP fields. For more information on external clock correction, refer to External Clock Synchronization .

Write: Normal Mode

Address: 0h base + 14h offset = 14h



FR_POCR field descriptions

Field	Description
0 WME	Write Mode External Correction. This bit controls the write mode of the EOC_AP and ERC_AP fields. 0 Write to EOC_AP and ERC_AP fields on register write. 1 No write to EOC_AP and ERC_AP fields on register write.
1–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–5 EOC_AP	External Offset Correction Application. This field is used to trigger the application of the external offset correction value defined in the Protocol Configuration Register 29 (FR_PCR29). 00 do not apply external offset correction value 01 reserved 10 subtract external offset correction value 11 add external offset correction value
6–7 ERC_AP	External Rate Correction Application. This field is used to trigger application of the external rate correction value defined in the Protocol Configuration Register 21 (FR_PCR21) 00 do not apply external rate correction value 01 reserved 10 subtract external rate correction value 11 add external rate correction value
8 BSY_WMC	**NOTE:** The name and function of this field varies depending on whether it is being read or written.

Table continues on the next page...

FR_POCR field descriptions (continued)

Field	Description
	BSY (Read-Only). Protocol Control Command Busy. This status bit indicates the acceptance of the protocol control command issued by the application via the POCCMD field. The CC sets this status bit when the application has issued a protocol control command via the POCCMD field. The CC clears this status bit when protocol control command was accepted by the PE. When the application issues a protocol control command while the BSY bit is asserted, the CC ignores this command, sets the protocol command ignored error flag PCMI_EF in the CHI Error Flag Register (FR_CHIERFR) , and will not change the value of the POCCMD field. WMC (Write-Only) Protocol Control Command Write. This status bit indicates the acceptance of the protocol control command issued by the application via the POCCMD field. The CC sets this status bit when the application has issued a protocol control command via the POCCMD field. The CC clears this status bit when protocol control command was accepted by the PE. When the application issues a protocol control command while the BSY bit is asserted, the CC ignores this command, sets the protocol command ignored error flag PCMI_EF in the CHI Error Flag Register (FR_CHIERFR) , and will not change the value of the POCCMD field. 0 (Write-Only) Write to POCCMD field on register write. 1 (Write-Only) Do not write to POCCMD field on register write. 0 (Read-Only) Command write idle, command accepted and ready to receive new protocol command. 1 (Read-Only) Command write busy, command not yet accepted, not ready to receive new protocol command.
9–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 POCCMD	Protocol Control Command. The application writes to this field to issue a protocol control command to the PE. The CC sends the protocol command to the PE immediately. While the transfer is running, the BSY bit is set. 0000 ALLOW_COLDSTART. Immediately activate capability of node to cold start cluster. 0001 ALL_SLOTS. Delayed transition to the all slots transmission mode. (Delayed means on completion of current communication cycle.) 0010 CONFIG. Immediately transition to the POC:config state. 0011 FREEZE. Immediately transition to the POC:halt state. 0100 READY, CONFIG_COMPLETE. Immediately transition to the POC:ready state. 0101 RUN. Immediately transition to the POC:startup start state. 0110 DEFAULT_CONFIG. Immediately transition to the POC:default config state. 0111 HALT. Delayed transition to the POC:halt state 1000 WAKEUP. Immediately initiate the wakeup procedure. 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Reserved

47.7.11 Global Interrupt Flag and Enable Register (FR_GIFER)

Write: Normal Mode

This register provides the means to control some of the interrupt request lines and provides the corresponding interrupt flags. The interrupt flags MIF, PRIF, CHIF, RBIF, and TBIF are the outcome of a binary OR of the related individual interrupt flags and interrupt enables . The generation scheme for these flags is depicted in Figure 47-33 . For more details on interrupt generation, see Interrupt Support . These flags are cleared automatically when all of the corresponding interrupt flags or interrupt enables in the related interrupt flag and enable registers are cleared by the application.

Address: 0h base + 16h offset = 16h

Bit	0	1	2	3	4	5	6	7
Read	MIF	PRIF	CHIF	WUPIF	FAFBIF	FAFAIF	RBIF	TBIF
Write				w1c	w1c	w1c		
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	MIE	PRIE	CHIE	WUPIE	FAFBIE	FAFAIE	RBIE	TBIE
Reset	0	0	0	0	0	0	0	0

FR_GIFER field descriptions

Field	Description
0 MIF	Module Interrupt Flag. This interrupt flag is set if at least one of the other interrupt flags in this register and the related interrupt enable bit are set. 0 No interrupt flag and related interrupt enable bit are set 1 At least one of the other interrupt flags in this register and the related interrupt bit are set.
1 PRIF	Protocol Interrupt Flag. This interrupt flag is set if at least one of the individual flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) and Protocol Interrupt Flag Register 1 (FR_PIFR1) and the related interrupt enable bit are set. 0 No individual protocol interrupt flag and related interrupt enable bit are set. 1 At least one of the individual protocol interrupt flags and the related interrupt enable bit are set.
2 CHIF	CHI Interrupt Flag. This interrupt flag is set if at least one of the error flags in the CHI Error Flag Register (FR_CHIERFR) and the chi error interrupt enable bit FR_GIFER[CHIE] are set . 0 All CHI error flags are equal to 0 or the chi error interrupt is disabled. 1 At least one CHI error flag and the chi error interrupt enable are is set.
3 WUPIF	Wakeup Interrupt Flag. This interrupt flag is set when the CC has received a wakeup symbol on the FlexRay bus. The application can determine on which channel the wakeup symbol was received by reading the related wakeup flags WUB and WUA in the Protocol Status Register 3 (FR_PSR3). 0 No Wakeup symbol received on FlexRay bus 1 Wakeup symbol received on FlexRay bus
4 FAFBIF	Receive FIFO Channel B Almost Full Interrupt Flag. This interrupt flag is set when one of the following events occurs a) the current number of FIFO B entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR), and the CC writes a received message into the FIFO B, or

Table continues on the next page...

FR_GIFER field descriptions (continued)

Field	Description
	b) the current number of FIFO B entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 no such event 1 FIFO B almost full event has occurred
5 FAFAIF	Receive FIFO Channel A Almost Full Interrupt Flag. This interrupt flag is set when one of the following events occurs a) the current number of FIFO A entries is equal to or greater than the watermark defined by the WM field in the Receive FIFO Watermark and Selection Register (FR_RFWMSR), and the CC writes a received message into the FIFO A, or b) the current number of FIFO B entries is at least 1 and the periodic timer as defined by Receive FIFO Periodic Timer Register (FR_RFPTR) expires. 0 no such event 1 FIFO A almost full event has occurred
6 RBIF	Receive Message Buffer Interrupt Flag. This interrupt flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) both the interrupt flag MBIF and the interrupt enable bit MBIE in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are asserted. The application can not clear this interrupt flag directly, instead it is cleared by the CC when all of the interrupt flags MBIF of the individual receive message buffers are cleared by the application or if the application has cleared the related interrupt enables bit MBIE. 0 None of the individual receive message buffers has the MBIF and MBIE flag set. 1 At least one individual receive message buffer has the MBIF and MBIE flag set.
7 TBIF	Transmit Message Buffer Interrupt Flag. This flag is set if for at least one of the individual message buffers (FR_MBCCSRn[MTD] = 1) both the interrupt flag MBIF and the interrupt enable bit MBIE in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) are equal to 1. The application can not clear this interrupt flag directly, instead, this interrupt flag is cleared by the CC when either all of the individual interrupt flags MBIF of the individual transmit message buffers are cleared by the application or the application has cleared the related interrupt enables bit MBIE. 0 None of the individual transmit message buffers has the MBIF and MBIE flag set. 1 At least one individual transmit message buffer has the MBIF and MBIE flag set.
8 MIE	Module Interrupt Enable. This bit controls if the Module Interrupt line is asserted when the MIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
9 PRIE	Protocol Interrupt Enable. This bit controls if the Protocol Interrupt line is asserted when the PRIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
10 CHIE	CHI Interrupt Enable. This bit controls if the CHI Interrupt line is asserted when the CHIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
11 WUPIE	Wakeup Interrupt Enable. This bit controls if the Wakeup Interrupt line is asserted when the WUPIF flag is set. 0 Disable interrupt line 1 Enable interrupt line

Table continues on the next page...

FR_GIFER field descriptions (continued)

Field	Description
12 FAFBIE	Receive FIFO Channel B Almost Full Interrupt Enable. This bit controls if the RX FIFO B Almost Full Interrupt line is asserted when the FAFBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
13 FAFAIE	Receive FIFO Channel A Almost Full Interrupt Enable. This bit controls if the RX FIFO A Almost Full Interrupt line is asserted when the FAFAIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
14 RBIE	Receive Message Buffer Interrupt Enable. This bit controls if the Receive Message Buffer Interrupt line is asserted when the RBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line
15 TBIE	Transmit Message Buffer Interrupt Enable. This bit controls if the Transmit Message Buffer Interrupt line is asserted when the TBIF flag is set. 0 Disable interrupt line 1 Enable interrupt line

47.7.12 Protocol Interrupt Flag Register 0 (FR_PIFR0)

Write: Normal Mode

The register holds one set of the protocol-related individual interrupt flags.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7
Read	FATL_IF	INTL_IF	ILCF_IF	CSA_IF	MRC_IF	MOC_IF	CCL_IF	MXS_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	MTX_IF	LTXB_IF	LTXA_IF	TBVB_IF	TBVA_IF	TI2_IF	TI1_IF	CYS_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

FR_PIFR0 field descriptions

Field	Description
0 FATL_IF	Fatal Protocol Error Interrupt Flag. This flag is set when the protocol engine has detected a fatal protocol error. In this case, the protocol engine goes into the POC:halt state immediately. The fatal protocol errors are: 1) pLatestTx violation, as described in the MAC process of the FlexRay protocol

Table continues on the next page...

FR_PIFR0 field descriptions (continued)

Field	Description
	2) transmission across slot boundary violation, as described in the FSP process of the FlexRay protocol 0 No such event. 1 Fatal protocol error detected.
1 INTL_IF	Internal Protocol Error Interrupt Flag. This flag is set when the protocol engine has detected an internal protocol error. In this case, the protocol engine goes into the POC:halt state immediately. An internal protocol error occurs when the protocol engine has not finished a calculation and a new calculation is requested. This can be caused by a hardware error. 0 No such event. 1 Internal protocol error detected.
2 ILCF_IF	Illegal Protocol Configuration Interrupt Flag. This flag is set when the protocol engine has detected an illegal protocol configuration parameter setting. In this case, the protocol engine goes into the POC:halt state immediately. The protocol engine checks the listen_timeout value programmed into the Protocol Configuration Register 14 (FR_PCR14) and Protocol Configuration Register 15 (FR_PCR15) when the CONFIG_COMPLETE command was sent by the application via the Protocol Operation Control Register (FR_POCR). If the value of listen_timeout is equal to zero, the protocol configuration setting is considered as illegal. 0 No such event. 1 Illegal protocol configuration detected.
3 CSA_IF	Cold Start Abort Interrupt Flag. This flag is set when the configured number of allowed cold start attempts is reached and none of these attempts was successful. The number of allowed cold start attempts is configured by the coldstart_attempts field in the Protocol Configuration Register 3 (FR_PCR3). 0 No such event. 1 Cold start aborted and no more coldstart attempts allowed.
4 MRC_IF	Missing Rate Correction Interrupt Flag. This flag is set when an insufficient number of measurements is available for rate correction at the end of the communication cycle. 0 No such event 1 Insufficient number of measurements for rate correction detected
5 MOC_IF	Missing Offset Correction Interrupt Flag. This flag is set when an insufficient number of measurements is available for offset correction. This is related to the MISSING_TERM event in the CSP process for offset correction in the FlexRay protocol. 0 No such event. 1 Insufficient number of measurements for offset correction detected.
6 CCL_IF	Clock Correction Limit Reached Interrupt Flag. This flag is set when the internal calculated offset or rate calculation values have reached or exceeded its configured thresholds as given by the offset_coorection_out field in the Protocol Configuration Register 9 (FR_PCR9) and the rate_correction_out field in the Protocol Configuration Register 14 (FR_PCR14). 0 No such event. 1 Offset or rate correction limit reached.
7 MXS_IF	Max Sync Frames Detected Interrupt Flag. This flag is set when the number of synchronization frames detected in the current communication cycle exceeds the value of the node_sync_max field in the Protocol Configuration Register 30 (FR_PCR30). **NOTE:** Only synchronization frames that have passed the synchronization frame acceptance and rejection filters are taken into account.

Table continues on the next page...

FR_PIFR0 field descriptions (continued)

Field	Description
	0 No such event. 1 More than node_sync_max sync frames detected.
8 MTX_IF	Media Access Test Symbol Received Interrupt Flag. This flag is set when the MTS symbol was received on channel A or channel B. 0 No such event. 1 MTS symbol received.
9 LTXB_IF	pLatestTx Violation on Channel B Interrupt Flag. This flag is set when the frame transmission on channel B in the dynamic segment exceeds the dynamic segment boundary. This is related to the pLatestTx violation, as described in the MAC process of the FlexRay protocol. 0 No such event. 1 pLatestTx violation occurred on channel B.
10 LTXA_IF	pLatestTx Violation on Channel A Interrupt Flag. This flag is set when the frame transmission on channel A in the dynamic segment exceeds the dynamic segment boundary. This is related to the pLatestTx violation as described in the MAC process of the FlexRay protocol. 0 No such event. 1 pLatestTx violation occurred on channel A.
11 TBVB_IF	Transmission across boundary on channel B Interrupt Flag. This flag is set when the frame transmission on channel B crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol. 0 No such event. 1 Transmission across boundary violation occurred on channel B.
12 TBVA_IF	Transmission across boundary on channel A Interrupt Flag. This flag is set when the frame transmission on channel A crosses the slot boundary. This is related to the transmission across slot boundary violation as described in the FSP process of the FlexRay protocol. 0 No such event. 1 Transmission across boundary violation occurred on channel A.
13 TI2_IF	Timer 2 Expired Interrupt Flag. This flag is set whenever timer 2 expires. 0 No such event. 1 Timer 2 has reached its time limit.
14 TI1_IF	Timer 1 Expired Interrupt Flag. This flag is set whenever timer 1 expires. 0 No such event 1 Timer 1 has reached its time limit
15 CYS_IF	Cycle Start Interrupt Flag. This flag is set when a communication cycle starts. 0 No such event 1 Communication cycle started.

47.7.13 Protocol Interrupt Flag Register 1 (FR_PIFR1)

Write: Normal Mode

The register holds one set of the protocol-related individual interrupt flags.

Address: 0h base + 1Ah offset = 1Ah

Bit	0	1	2	3	4	5	6	7
Read	EMC_IF	IPC_IF	PECF_IF	PSC_IF	SSI3_IF	SSI2_IF	SSI1_IF	SSI0_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		EVT_IF	ODT_IF	0			
Write			w1c	w1c				
Reset	0	0	0	0	0	0	0	0

FR_PIFR1 field descriptions

Field	Description
0 EMC_IF	Error Mode Changed Interrupt Flag. This flag is set when the value of the ERRMODE bit field in the Protocol Status Register 0 (FR_PSR0) is changed by the CC. 0 No such event. 1 ERRMODE field changed.
1 IPC_IF	Illegal Protocol Control Command Interrupt Flag. This flag is set when the PE tries to execute a protocol control command, which was issued via the POCCMD field of the Protocol Operation Control Register (FR_POCR), and detects that this protocol control command is not allowed in the current protocol state. In this case the command is not executed. For more details, see Protocol Control Command Execution . 0 No such event. 1 Illegal protocol control command detected.
2 PECF_IF	Protocol Engine Communication Failure Interrupt Flag. This flag is set if the CC has detected a communication failure between the PE and the CHI. 0 No such event. 1 Protocol Engine Communication Failure detected.
3 PSC_IF	Protocol State Changed Interrupt Flag. This flag is set when the protocol state in the PROTSTATE field in the Protocol Status Register 0 (FR_PSR0) has changed. 0 No such event. 1 Protocol state changed.
4 SSI3_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
5 SSI2_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
6 SSI1_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.

Table continues on the next page...

FR_PIFR1 field descriptions (continued)

Field	Description
7 SSI0_IF	Slot Status Counter Incremented Interrupt Flag. Each of these flags is set when the SLOTSTATUSCNT field in the corresponding Slot Status Counter Registers (FR_SSCRn) is incremented. 0 No such event. 1 The corresponding slot status counter has incremented.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 EVT_IF	Even Cycle Table Written Interrupt Flag. This flag is set if the CC has written the sync frame measurement / ID tables into the FlexRay memory area for the even cycle. 0 No such event. 1 Sync frame measurement table written
11 ODT_IF	Odd Cycle Table Written Interrupt Flag. This flag is set if the CC has written the sync frame measurement / ID tables into the FlexRay memory area for the odd cycle. 0 No such event. 1 Sync frame measurement table written
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.7.14 Protocol Interrupt Enable Register 0 (FR_PIER0)

Write: Anytime

This register defines whether or not the individual interrupt flags defined in the Protocol Interrupt Flag Register 0 (FR_PIFR0) can generate a protocol interrupt request.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7
Read Write	FATL_IE	INTL_IE	ILCF_IE	CSA_IE	MRC_IE	MOC_IE	CCL_IE	MXS_IE
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	MTX_IE	LTXB_IE	LTXA_IE	TBVB_IE	TBVA_IE	TI2_IE	TI1_IE	CYS_IE
Reset	0	0	0	0	0	0	0	0

FR_PIER0 field descriptions

Field	Description
0 FATL_IE	Fatal Protocol Error Interrupt Enable. This bit controls FATL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
1 INTL_IE	Internal Protocol Error Interrupt Enable. This bit controls INTL_IF interrupt request generation.

Table continues on the next page...

FR_PIER0 field descriptions (continued)

Field	Description
	0 interrupt request generation disabled 1 interrupt request generation enabled
2 ILCF_IE	Illegal Protocol Configuration Interrupt Enable. This bit controls ILCF_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
3 CSA_IE	Cold Start Abort Interrupt Enable. This bit controls CSA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
4 MRC_IE	Missing Rate Correction Interrupt Enable. This bit controls MRC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
5 MOC_IE	Missing Offset Correction Interrupt Enable. This bit controls MOC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
6 CCL_IE	Clock Correction Limit Reached Interrupt Enable. This bit controls CCL_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
7 MXS_IE	Max Sync Frames Detected Interrupt Enable. This bit controls MXS_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
8 MTX_IE	Media Access Test Symbol Received Interrupt Enable. This bit controls MTX_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
9 LTXB_IE	pLatestTx Violation on Channel B Interrupt Enable. This bit controls LTXB_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
10 LTXA_IE	pLatestTx Violation on Channel A Interrupt Enable. This bit controls LTXA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
11 TBVB_IE	Transmission across boundary on channel B Interrupt Enable. This bit controls TBVB_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
12 TBVA_IE	Transmission across boundary on channel A Interrupt Enable. This bit controls TBVA_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

Table continues on the next page...

FR_PIER0 field descriptions (continued)

Field	Description
13 TI2_IE	Timer 2 Expired Interrupt Enable. This bit controls TI1_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
14 TI1_IE	Timer 1 Expired Interrupt Enable. This bit controls TI1_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
15 CYS_IE	Cycle Start Interrupt Enable. This bit controls CYC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

47.7.15 Protocol Interrupt Enable Register 1 (FR_PIER1)

Write: Anytime

This register defines whether or not the individual interrupt flags defined in Protocol Interrupt Flag Register 1 (FR_PIFR1) can generate a protocol interrupt request.

Address: 0h base + 1Eh offset = 1Eh

Bit	0	1	2	3	4	5	6	7
Read Write	EMC_IE	IPC_IE	PECF_IE	PSC_IE	SSI3_IE	SSI2_IE	SSI1_IE	SSI0_IE
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		EVT_IE	ODT_IE	0			
Write								
Reset	0	0	0	0	0	0	0	0

FR_PIER1 field descriptions

Field	Description
0 EMC_IE	Error Mode Changed Interrupt Enable. This bit controls EMC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
1 IPC_IE	Illegal Protocol Control Command Interrupt Enable. This bit controls IPC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
2 PECF_IE	Protocol Engine Communication Failure Interrupt Enable. This bit controls PECF_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled

Table continues on the next page...

FR_PIER1 field descriptions (continued)

Field	Description
3 PSC_IE	Protocol State Changed Interrupt Enable. This bit controls PSC_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
4 SSI3_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
5 SSI2_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
6 SSI1_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
7 SSI0_IE	Slot Status Counter Incremented Interrupt Enable. This bit controls SSI[3:0]_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10 EVT_IE	Even Cycle Table Written Interrupt Enable. This bit controls EVT_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
11 ODT_IE	Odd Cycle Table Written Interrupt Enable. This bit controls ODT_IF interrupt request generation. 0 interrupt request generation disabled 1 interrupt request generation enabled
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.7.16 CHI Error Flag Register (FR_CHIERFR)

Write: Normal Mode

This register holds the CHI related error flags. The interrupt generation for each of these error flags is controlled by the CHI interrupt enable bit CHIE in the Global Interrupt Flag and Enable Register (FR_GIFER).

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7
Read	FRLB_EF	FRLA_EF	PCMI_EF	FOVB_EF	FOVA_EF	MBS_EF	MBU_EF	LCK_EF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0	SBCF_EF	FID_EF	DPL_EF	SPL_EF	NML_EF	NMF_EF	ILSA_EF
Write		w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

FR_CHIERFR field descriptions

Field	Description
0 FRLB_EF	Frame Lost Channel B Error Flag. This flag is set if a complete frame was received on channel B but could not be stored in the selected individual message buffer because this message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost. 0 No such event 1 Frame lost on channel B detected
1 FRLA_EF	Frame Lost Channel A Error Flag. This flag is set if a complete frame was received on channel A but could not be stored in the selected individual message buffer because this message buffer is currently locked by the application. In this case, the frame and the related slot status information are lost. 0 No such error 1 Frame lost on channel A detected
2 PCMI_EF	Protocol Command Ignored Error Flag. This flag is set if the application has issued a POC command by writing to the POCCMD field in the Protocol Operation Control Register (FR_POCR) while the BSY flag is equal to 1. In this case the command is ignored by the CC and is lost. 0 No such error 1 POC command ignored
3 FOVB_EF	Receive FIFO Overrun Channel B Error Flag. This flag is set when an overrun of the FIFO for channel B occurred. This error occurs if a semantically valid frame was received on channel B and matches the all criteria to be appended to the FIFO for channel B but the FIFO is full. In this case, the received frame and its related slot status information is lost. 0 No such error 1 FIFO overrun on channel B has been detected
4 FOVA_EF	Receive FIFO Overrun Channel A Error Flag. This flag is set when an overrun of the FIFO for channel A occurred. This error occurs if a semantically valid frame was received on channel A and matches the all criteria to be appended to the FIFO for channel A but the FIFO is full. In this case, the received frame and its related slot status information is lost. 0 No such error 1 FIFO overrun on channel B has been detected
5 MBS_EF	Message Buffer Search Error Flag. This flag is set if at least one of the following events occurs: a) The message buffer search engine is still running while the next search must be started due to the FlexRay protocol timing. b) A message buffer index greater than 131 is detected in the FR_MBIDXR[MBIDX] field of an found message buffer or in one of the FR_RSBIR[RSBIDX] fields.

Table continues on the next page...

FR_CHIERFR field descriptions (continued)

Field	Description
	Refer to Message Buffer Search Error for details. 0 No such event 1 Search engine active while search start appears or illegal message buffer index detected
6 MBU_EF	Message Buffer Utilization Error Flag. This flag is asserted if the application writes to a message buffer control field that is beyond the number of utilized message buffers programmed in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR). If the application writes to a FR_MBCCSRn register with n > LAST_MB_UTIL, the CC ignores the write attempt and asserts the message buffer utilization error flag MBU_EF in the CHI Error Flag Register (FR_CHIERFR). 0 No such event 1 Non-utilized message buffer enabled
7 LCK_EF	Lock Error Flag. This flag is set if the application tries to lock a message buffer that is already locked by the CC due to internal operations. In that case, the CC does not grant the lock to the application. The application must issue the lock request again. 0 No such error 1 Lock error detected
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 SBCF_EF	System Bus Communication Failure Error Flag. This flag is set if a system bus access was not finished within the required amount of time (see System Bus Access Timeout). 0 No such event 1 System bus access not finished in time
10 FID_EF	Frame ID Error Flag. This flag is set if the frame ID stored in the message buffer header area differs from the frame ID stored in the message buffer control register and "also an illegal CHI command RUN is issued in POC:normal active state". 0 No such error occurred 1 Frame ID error occurred
11 DPL_EF	Dynamic Payload Length Error Flag. This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the dynamic segment is greater than the maximum payload length for the dynamic segment as it is configured in the corresponding protocol configuration register field max_payload_length_dynamic in the Protocol Configuration Register 24 (FR_PCR24). 0 No such error occurred 1 Dynamic payload length error occurred
12 SPL_EF	Static Payload Length Error Flag. This flag is set if the payload length written into the message buffer header field of a transmit message buffer assigned to the static segment is different from the payload length for the static segment as it is configured in the corresponding protocol configuration register field payload_length_static in the Protocol Configuration Register 19 (FR_PCR19). 0 No such error occurred 1 Static payload length error occurred
13 NML_EF	Network Management Length Error Flag. This flag is set if the payload length written into the header structure of a receive message buffer assigned to the static segment is less than the configured length of the Network Management Vector as configured in the Network Management Vector Length Register (FR_NMVLR). In this case the received part of the Network Management Vector will be used to update the Network Management Vector.

Table continues on the next page...

FR_CHIERFR field descriptions (continued)

Field	Description
	0 No such error occurred 1 Network management length error occurred
14 NMF_EF	Network Management Frame Error Flag. This flag is set if a received message in the static segment with a Preamble Indicator flag PP asserted has its Null Frame indicator flag NF asserted as well. In this case, the Global Network Management Registers (see Network Management Vector Registers (FR_NMVR0-FR_NMVR5)) are not updated. 0 No such error occurred 1 Network management frame error occurred
15 ILSA_EF	Illegal System Bus Address Error Flag. This flag is set if the external system bus subsystem has detected an access to an illegal system bus address from the CC (see System Bus Illegal Address Access). 0 No such event 1 Illegal system bus address accessed

47.7.17 Message Buffer Interrupt Vector Register (FR_MBIVEC)

This register indicates the lowest numbered receive message buffer and the lowest numbered transmit message buffer that have their interrupt status flag MBIF and interrupt enable MBIE bits asserted. This means that message buffers with lower message buffer numbers have higher priority.

Address: 0h base + 22h offset = 22h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	TBIVEC							0	RBIVEC						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MBIVEC field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 TBIVEC	Transmit Buffer Interrupt Vector. This field provides the number of the lowest numbered enabled transmit message buffer that has its interrupt status flag MBIF and its interrupt enable bit MBIE set. If there is no transmit message buffer with the interrupt status flag MBIF and the interrupt enable MBIE bits asserted, the value in this field is set to 0.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 RBIVEC	Receive Buffer Interrupt Vector. This field provides the message buffer number of the lowest numbered receive message buffer which has its interrupt flag MBIF and its interrupt enable bit MBIE asserted. If there is no receive message buffer with the interrupt status flag MBIF and the interrupt enable MBIE bits asserted, the value in this field is set to 0.

47.7.18 Channel A Status Error Counter Register (FR_CASERCR)

This register provides the channel status error counter for channel A. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the NIT. The slot status vector contains the four protocol related error indicator bits vSS!SyntaxError, vSS!ContentError, vSS!BViolation, and vSS!TxConflict. The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring, see Slot Status Monitoring.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CHAERSCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_CASERCR field descriptions

Field	Description
0–15 CHAERSCNT	Channel A Status Error Counter. This field provides the current value channel status error counter. The counter value is updated within the first macrotick of the following slot or segment.

47.7.19 Channel B Status Error Counter Register (FR_CBSERCR)

This register provides the channel status error counter for channel B. The protocol engine generates a slot status vector for each static slot, each dynamic slot, the symbol window, and the NIT. The slot status vector contains the four protocol related error indicator bits vSS!SyntaxError, vSS!ContentError, vSS!BViolation, and vSS!TxConflict. The CC increments the status error counter by 1 if, for a slot or segment, at least one error indicator bit is set to 1. The counter wraps around after it has reached the maximum value. For more information on slot status monitoring see Slot Status Monitoring.

Address: 0h base + 26h offset = 26h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	CHBERSCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_CBSERCR field descriptions

Field	Description
0–15 CHBERSCNT	Channel B Status Error Counter. This field provides the current channel status error count. The counter value is updated within the first macrotick of the following slot or segment.

47.7.20 Protocol Status Register 0 (FR_PSR0)

This register provides information about the current protocol status.

Address: 0h base + 28h offset = 28h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	ERRMODE		SLOTMODE		0	PROTSTATE			STARTUPSTATE				0	WAKEUPSTATUS		
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PSR0 field descriptions

Field	Description
0–1 ERRMODE	Error Mode. protocol related variable: vPOC!ErrorMode. This field indicates the error mode of the protocol. 00 ACTIVE 01 PASSIVE 10 COMM_HALT 11 reserved
2–3 SLOTMODE	Slot Mode. protocol related variable: vPOC!SlotMode. This field indicates the slot mode of the protocol. 00 SINGLE 01 ALL_PENDING 10 ALL 11 reserved
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–7 PROTSTATE	Protocol State. protocol related variable: vPOC!State. This field indicates the state of the protocol. 000 POC:default config 001 POC:config 010 POC:wakeup 011 POC:ready 100 POC:normal passive 101 POC:normal active 110 POC:halt 111 POC:startup

Table continues on the next page...

FR_PSR0 field descriptions (continued)

Field	Description
8–11 STARTUPSTATE	Startup State. protocol related variable: vPOC!StartupState. This field indicates the current sub-state of the startup procedure. 0000 reserved 0001 reserved 0010 POC:coldstart collision resolution 0011 POC:coldstart listen 0100 POC:integration consistency check 0101 POC:integration listen 0110 reserved 0111 POC:initialize schedule 1000 reserved 1001 reserved 1010 POC:coldstart consistency check 1011 reserved 1100 reserved 1101 POC:integration coldstart check 1110 POC:coldstart gap 1111 POC:coldstart join
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–15 WAKEUPSTATUS	Wakeup Status. protocol related variable: vPOC!WakeupStatus. This field provides the outcome of the execution of the wakeup mechanism. 000 UNDEFINED 001 RECEIVED_HEADER 010 RECEIVED_WUP 011 COLLISION_ HEADER 100 COLLISION_WUP 101 COLLISION_UNKNOWN 110 TRANSMITTED 111 reserved

47.7.21 Protocol Status Register 1 (FR_PSR1)

Write: Normal Mode

Address: 0h base + 2Ah offset = 2Ah

Bit	0	1	2	3	4	5	6	7
Read	CSAA	CSP	0	REMCSAT				
Write	w1c							
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	CPN	HHR	FRZ	APTAC				
Write								
Reset	0	0	0	0	0	0	0	0

FR_PSR1 field descriptions

Field	Description
0 CSAA	Cold Start Attempt Aborted Flag. protocol related event: 'set coldstart abort indicator in CHI' This flag is set when the CC has aborted a cold start attempt. 0 No such event 1 Cold start attempt aborted
1 CSP	Leading Cold Start Path. This status bit is set when the CC has reached the POC:normal active state via the leading cold start path. This indicates that this node has started the network 0 No such event 1 POC:normal active reached from POC:startup state via leading cold start path
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 REMCSAT	Remaining Coldstart Attempts. protocol related variable: vRemainingColdstartAttempts This field provides the number of remaining cold start attempts that the CC will execute.
8 CPN	Leading Cold Start Path Noise. protocol related variable: vPOC!ColdstartNoise This status bit is set if the CC has reached the POC:normal active state via the leading cold start path under noise conditions. This indicates there was some activity on the FlexRay bus while the CC was starting up the cluster. 0 No such event 1 POC:normal active state was reached from POC:startup state via noisy leading cold start path
9 HHR	Host Halt Request Pending. protocol related variable: vPOC!CHIHaltRequest This status bit is set when CC receives the HALT command from the application via the Protocol Operation Control Register (FR_POCR). The CC clears this status bit after a hard reset condition or when the protocol is in the POC:default config state. 0 No such event 1 HALT command received

Table continues on the next page...

FR_PSR1 field descriptions (continued)

Field	Description
10 FRZ	Freeze Occurred. protocol related variable: vPOC!Freeze This status bit is set when the CC has reached the POC:halt state due to the host FREEZE command or due to an internal error condition requiring immediate halt. The CC clears this status bit after a hard reset condition or when the protocol is in the POC:default config state. 0 No such event 1 Immediate halt due to FREEZE or internal error condition
11–15 APTAC	Allow Passive to Active Counter. protocol related variable: vPOC!vAllowPassivetoActive This field provides the number of consecutive even/odd communication cycle pairs that have passed with valid rate and offset correction terms, but the protocol is still in the POC:normal passive state due to an application configured delay to enter POC:normal active state. This delay is defined by the allow_passive_to_active field in the Protocol Configuration Register 12 (FR_PCR12).

47.7.22 Protocol Status Register 2 (FR_PSR2)

This register provides a snapshot of status information about the Network Idle Time NIT, the Symbol Window and the clock synchronization. The NIT related status bits NBVB, NSEB, NBVA, and NSEA are updated by the CC after the end of the NIT and before the end of the first slot of the next communication cycle. The Symbol Window related status bits STCB, SBVB, SSEB, MTB, STCA, SBVA, SSEB, and MTA are updated by the CC after the end of the symbol window and before the end of the current communication cycle. If no symbol window is configured, the symbol window related status bits remain in their reset state. The clock synchronization related CLKCORRFAILCNT is updated by the CC after the end of the static segment and before the end of the current communication cycle.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7
Read	NBVB	NSEB	STCB	SBVB	SSEB	MTB	NBVA	NSEA
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	STCA	SBVA	SSEA	MTA	CKCORFCNT			
Write								
Reset	0	0	0	0	0	0	0	0

FR_PSR2 field descriptions

Field	Description
0 NBVB	NIT Boundary Violation on Channel B. protocol related variable: vSS!BViolation for NIT on channel B This status bit is set when there was some media activity on the FlexRay bus channel B at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
1 NSEB	NIT Syntax Error on Channel B. protocol related variable: vSS!SyntaxError for NIT on channel B This status bit is set when a syntax error was detected during NIT on channel B. 0 No such event 1 Syntax error detected
2 STCB	Symbol Window Transmit Conflict on Channel B. protocol related variable: vSS!TxConflict for symbol window on channel B This status bit is set if there was a transmission conflict during the symbol window on channel B. 0 No such event 1 Transmission conflict detected
3 SBVB	Symbol Window Boundary Violation on Channel B. protocol related variable: vSS!BViolation for symbol window on channel B This status bit is set if there was some media activity on the FlexRay bus channel B at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected
4 SSEB	Symbol Window Syntax Error on Channel B. protocol related variable: vSS!SyntaxError for symbol window on channel B This status bit is set when a syntax error was detected during the symbol window on channel B. 0 No such event 1 Syntax error detected
5 MTB	Media Access Test Symbol MTS Received on Channel B. protocol related variable: vSS!ValidMTS for Symbol Window on channel B This status bit is set if the Media Access Test Symbol MTS was received in the symbol window on channel B. 0 No such event 1 MTS symbol received
6 NBVA	NIT Boundary Violation on Channel A. protocol related variable: vSS!BViolation for NIT on channel A This status bit is set when there was some media activity on the FlexRay bus channel A at the end of the NIT. 0 No such event 1 Media activity at boundaries detected
7 NSEA	NIT Syntax Error on Channel A. protocol related variable: vSS!SyntaxError for NIT on channel A This status bit is set when a syntax error was detected during NIT on channel A. 0 No such event 1 Syntax error detected

Table continues on the next page...

FR_PSR2 field descriptions (continued)

Field	Description
8 STCA	Symbol Window Transmit Conflict on Channel A. protocol related variable: vSS!TxConflict for symbol window on channel A This status bit is set if there was a transmission conflicts during the symbol window on channel A. 0 No such event 1 Transmission conflict detected
9 SBVA	Symbol Window Boundary Violation on Channel A. protocol related variable: vSS!BViolation for symbol window on channel A This status bit is set if there was some media activity on the FlexRay bus channel A at the start or at the end of the symbol window. 0 No such event 1 Media activity at boundaries detected
10 SSEA	Symbol Window Syntax Error on Channel A. protocol related variable: vSS!SyntaxError for symbol window on channel A This status bit is set when a syntax error was detected during the symbol window on channel A. 0 No such event 1 Syntax error detected
11 MTA	Media Access Test Symbol MTS Received on Channel A. protocol related variable: vSS!ValidMTS for symbol window on channel A This status bit is set if the Media Access Test Symbol MTS was received in the symbol window on channel A. 1 MTS symbol received 0 No such event
12–15 CKCORFCNT	Clock Correction Failed Counter. protocol related variable: vClockCorrectionFailed This field provides the number of consecutive even/odd communication cycle pairs that have passed without clock synchronization having performed an offset or a rate correction due to lack of synchronization frames. It is not incremented when it has reached the configured value of either max_without_clock_correction_fatal or max_without_clock_correction_passive as defined in the Protocol Configuration Register 8 (FR_PCR8). The CC resets this counter on a hard reset condition, when the protocol enters the POC:normal active state, or when both the rate and offset correction terms have been calculated successfully.

47.7.23 Protocol Status Register 3 (FR_PSR3)

Write: Normal Mode

This register provides aggregated channel status information as an accrued status of channel activity for all communication slots, regardless of whether they are assigned for transmission or subscribed for reception. It provides accrued information for the symbol window, the NIT, and the wakeup status.

Address: 0h base + 2Eh offset = 2Eh

Bit	0	1	2	3	4	5	6	7
Read	0		WUB	ABVB	AACB	ACEB	ASEB	AVFB
Write			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0		WUA	ABVA	AACA	ACEA	ASEA	AVFA
Write			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

FR_PSR3 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 WUB	Wakeup Symbol Received on Channel B. This flag is set when a wakeup symbol was received on channel B. 0 No wakeup symbol received 1 Wakeup symbol received
3 ABVB	Aggregated Boundary Violation on Channel B. This flag is set when a boundary violation has been detected on channel B. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected
4 AACB	Aggregated Additional Communication on Channel B. This flag is set when at least one valid frame was received on channel B in a slot that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
5 ACEB	Aggregated Content Error on Channel B. This flag is set when a content error has been detected on channel B. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected
6 ASEB	Aggregated Syntax Error on Channel B. This flag is set when a syntax error has been detected on channel B. Syntax errors are detected in the communication slots, the symbol window and the NIT. 0 No syntax error detected 1 Syntax errors detected
7 AVFB	Aggregated Valid Frame on Channel B. This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel B. 1 At least one syntactically valid frame received 0 No syntactically valid frames received
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_PSR3 field descriptions (continued)

Field	Description
10 WUA	Wakeup Symbol Received on Channel A. This flag is set when a wakeup symbol was received on channel A. 0 No wakeup symbol received 1 Wakeup symbol received
11 ABVA	Aggregated Boundary Violation on Channel A. This flag is set when a boundary violation has been detected on channel A. Boundary violations are detected in the communication slots, the symbol window, and the NIT. 0 No boundary violation detected 1 Boundary violation detected
12 AACA	Aggregated Additional Communication on Channel A. This flag is set when a valid frame was received in a slot on channel A that also contained an additional communication with either syntax error, content error, or boundary violations. 0 No additional communication detected 1 Additional communication detected
13 ACEA	Aggregated Content Error on Channel A. This flag is set when a content error has been detected on channel A. Content errors are detected in the communication slots, the symbol window, and the NIT. 0 No content error detected 1 Content error detected
14 ASEA	Aggregated Syntax Error on Channel A. This flag is set when a syntax error has been detected on channel A. Syntax errors are detected in the communication slots, the symbol window, and the NIT. 0 No syntax error detected 1 Syntax errors detected
15 AVFA	Aggregated Valid Frame on Channel A. This flag is set when a syntactically correct valid frame has been received in any static or dynamic slot through channel A. 0 No syntactically valid frames received 1 At least one syntactically valid frame received

47.7.24 Macrotick Counter Register (FR_MTCTR)

This register provides the macrotick count of the current communication cycle.

Address: 0h base + 30h offset = 30h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		MTCT													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MTCTR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 MTCT	Macrotick Counter. protocol related variable: vMacrotick This field provides the macrotick count of the current communication cycle.

47.7.25 Cycle Counter Register (FR_CYCTR)

This register provides the number of the current communication cycle.

Address: 0h base + 32h offset = 32h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0										CYCCNT					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_CYCTR field descriptions

Field	Description
0–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 CYCCNT	Cycle Counter. protocol related variable: vCycleCounter This field provides the number of the current communication cycle. If the counter reaches the maximum value of 63, the counter wraps and starts from zero again.

47.7.26 Slot Counter Channel A Register (FR_SLTCTAR)

This register provides the number of the current slot in the current communication cycle for channel A.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					SLOTCNTA										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SLTCTAR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 SLOTCNTA	Slot Counter Value for Channel A. protocol related variable: vSlotCounter for channel A This field provides the number of the current slot in the current communication cycle.

47.7.27 Slot Counter Channel B Register (FR_SLTCTBR)

This register provides the number of the current slot in the current communication cycle for channel B.

Address: 0h base + 36h offset = 36h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					SLOTCNTB										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SLTCTBR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 SLOTCNTB	Slot Counter Value for Channel B. protocol related variable: vSlotCounter for channel B This field provides the number of the current slot in the current communication cycle.

47.7.28 Rate Correction Value Register (FR_RTCORVR)

This register provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT of each odd numbered communication cycle.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	RATECORR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RTCORVR field descriptions

Field	Description
0–15 RATECORR	Rate Correction Value. protocol related variable: vRateCorrection (before value limitation and external rate correction) This field provides the sign extended rate correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external rate correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by rate_correction_out in the Protocol Configuration Register 13 (FR_PCR13), the clock correction reached limit interrupt flag CCL_IF is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0). **NOTE:** If the CC was not able to calculate a new rate correction term due to a lack of synchronization frames, the RATECORR value is not updated.

47.7.29 Offset Correction Value Register (FR_OFCORVR)

This register provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The CC updates this register during the NIT.

Address: 0h base + 3Ah offset = 3Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	OFFSETCORR															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_OFCORVR field descriptions

Field	Description
0–15 OFFSETCORR	Offset Correction Value. protocol related variable: vOffsetCorrection (before value limitation and external offset correction) This field provides the sign extended offset correction value in microticks as it was calculated by the clock synchronization algorithm. The value is represented in 2's complement format. This value does not include the value limitation and the application of the external offset correction. If the magnitude of the internally calculated rate correction value exceeds the limit given by offset_correction_out field in the Protocol Configuration Register 29 (FR_PCR29), the clock correction reached limit interrupt flag CCL_IF is set in the Protocol Interrupt Flag Register 0 (FR_PIFR0). **NOTE:** If the CC was not able to calculate an new offset correction term due to a lack of synchronization frames, the OFFSETCORR value is not updated.

47.7.30 Combined Interrupt Flag Register (FR_CIFR)

This register provides five combined interrupt flags and a copy of three individual interrupt flags. The combined interrupt flags are the result of a binary OR of the values of other interrupt flags regardless of the state of the interrupt enable bits. The generation scheme for the combined interrupt flags is depicted in Figure 47-35 . The individual interrupt flags WUPIF, FAFBIF, and FAFAIF are copies of corresponding flags in the Global Interrupt Flag and Enable Register (FR_GIFER) and are provided here to simplify the application interrupt flag check. To clear the individual interrupt flags, the application must use the Global Interrupt Flag and Enable Register (FR_GIFER).

NOTE

The meanings of the combined status bits MIF, PRIF, CHIF, RBIF, and TBIF are different from those mentioned in the Global Interrupt Flag and Enable Register (FR_GIFER).

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7
Read	0							
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	MIF	PRIF	CHIF	WUPIF	FAFBIF	FAFAIF	RBIF	TBIF
Write								
Reset	0	0	0	0	0	0	0	0

FR_CIFR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 MIF	Module Interrupt Flag. This flag is set if there is at least one interrupt source that has its interrupt flag asserted. 0 No interrupt source has its interrupt flag asserted 1 At least one interrupt source has its interrupt flag asserted
9 PRIF	Protocol Interrupt Flag. This flag is set if at least one of the individual protocol interrupt flags in the Protocol Interrupt Flag Register 0 (FR_PIFR0) or Protocol Interrupt Flag Register 1 (FR_PIFR1) is equal to 1. 0 All individual protocol interrupt flags are equal to 0 1 At least one of the individual protocol interrupt flags is equal to 1
10 CHIF	CHI Interrupt Flag. This flag is set if at least one of the individual CHI error flags in the CHI Error Flag Register (FR_CHIERFR) is equal to 1.

Table continues on the next page...

FR_CIFR field descriptions (continued)

Field	Description
	0 All CHI error flags are equal to 0 1 At least one CHI error flag is equal to 1
11 WUPIF	Wakeup Interrupt Flag. Provides the same value as FR_GIFER[WUPIF]
12 FAFBIF	Receive FIFO Channel B Almost Full Interrupt Flag. Provides the same value as FR_GIFER[FAFBIF]
13 FAFAIF	Receive FIFO Channel A Almost Full Interrupt Flag. Provides the same value as FR_GIFER[FAFAIF]
14 RBIF	Receive Message Buffer Interrupt Flag. This flag is set if for at least one of the individual receive message buffers (FR_MBCCSRn[MTD] = 0) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual receive message buffers has the MBIF flag asserted. 1 At least one individual receive message buffers has the MBIF flag asserted.
15 TBIF	Transmit Message Buffer Interrupt Flag. This flag is set if for at least one of the individual transmit message buffers (FR_MBCCSRn[MTD] = 1) the interrupt flag MBIF in the corresponding Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) is equal to 1. 0 None of the individual transmit message buffers has the MBIF flag asserted. 1 At least one individual transmit message buffers has the MBIF flag asserted.

47.7.31 System Memory Access Time-Out Register (FR_SYMATOR)

Write: Disabled Mode

Address: 0h base + 3Eh offset = 3Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								TIMEOUT							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

FR_SYMATOR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 TIMEOUT	System Memory Access Time-Out. This value defines when a system bus access timout is detected. For a detailed description see Configure System Memory Access Time-Out Register (FR_SYMATOR) and System Bus Access Timeout .

47.7.32 Sync Frame Counter Register (FR_SFCNTR)

This register provides the number of synchronization frames that are used for clock synchronization in the last even and in the last odd numbered communication cycle. This register is updated after the start of the NIT and before 10 MT after offset correction start.

NOTE

If the application has locked the even synchronization table at the end of the static segment of an even communication cycle, the CC will not update the fields SFEVB and SFEVA.

NOTE

If the application has locked the odd synchronization table at the end of the static segment of an odd communication cycle, the CC will not update the values SFODB and SFODA.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SFEVB				SFEVA				SFODB				SFODA			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFCNTR field descriptions

Field	Description
0–3 SFEVB	Sync Frames Channel B, even cycle. protocol related variable: size of (vsSyncIdListB for even cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
4–7 SFEVA	Sync Frames Channel A, even cycle. protocol related variable: size of (vsSyncIdListA for even cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization.
8–11 SFODB	Sync Frames Channel B, odd cycle. protocol related variable: size of (vsSyncIdListB for odd cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization .
12–15 SFODA	Sync Frames Channel A, odd cycle. protocol related variable: size of (vsSyncIdListA for odd cycle) This field provides the size of the internal list of frame IDs of received synchronization frames used for clock synchronization .

47.7.33 Sync Frame Table Offset Register (FR_SFTOR)

Write: POC:config

This register defines the FlexRay memory area related offset for sync frame tables. For more details, see Sync Frame ID and Sync Frame Deviation Tables .

Address: 0h base + 42h offset = 42h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SFT_OFFSET															0
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFTOR field descriptions

Field	Description
0–14 SFT_OFFSET	Sync Frame Table Offset. The offset of the Sync Frame Tables in the FlexRay memory area. This offset is required to be 16-bit aligned. Thus STF_OFFSET[0] is always 0.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.7.34 Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR)

Write: Normal Mode

This register provides configuration, control, and status information related to the generation and access of the clock sync ID tables and clock sync measurement tables. For a detailed description, see Sync Frame ID and Sync Frame Deviation Tables .

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7
Read	0	0	CYCNUM					
Write	ELKT	OLKT						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	ELKS	OLKS	EVAL	OVAL	0	0	SDVEN	SIDEN
Write						OPT		
Reset	0	0	0	0	0	0	0	0

FR_SFTCCSR field descriptions

Field	Description
0 ELKT	Even Cycle Tables Lock/Unlock Trigger. This trigger bit is used to lock and unlock the even cycle tables. 0 No effect 1 Triggers lock/unlock of the even cycle tables.
1 OLKT	Odd Cycle Tables Lock/Unlock Trigger. This trigger bit is used to lock and unlock the odd cycle tables. 0 No effect 1 Triggers lock/unlock of the odd cycle tables.

Table continues on the next page...

FR_SFTCCSR field descriptions (continued)

Field	Description
2–7 CYCNUM	Cycle Number. This field provides the number of the cycle in which the currently locked table was recorded. If none or both tables are locked, this value is related to the even cycle table.
8 ELKS	Even Cycle Tables Lock Status. This status bit indicates whether the application has locked the even cycle tables. 0 Application has not locked the even cycle tables. 1 Application has locked the even cycle tables.
9 OLKS	Odd Cycle Tables Lock Status. This status bit indicates whether the application has locked the odd cycle tables. 0 Application has not locked the odd cycle tables. 1 Application has locked the odd cycle tables.
10 EVAL	Even Cycle Tables Valid. This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the even cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent).
11 OVAL	Odd Cycle Tables Valid. This status bit indicates whether the Sync Frame ID and Sync Frame Deviation Tables for the odd cycle are valid. The CC clears this status bit when it starts updating the tables, and sets this bit when it has finished the table update. 0 Tables are not valid (update is ongoing) 1 Tables are valid (consistent).
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 OPT	One Pair Trigger. This trigger bit controls whether the CC writes continuously or only one pair of Sync Frame Tables into the FlexRay memory area. If this trigger is set to 1 while SDVEN or SIDEN is set to 1, the CC writes only one pair of the enabled Sync Frame Tables corresponding to the next even-odd-cycle pair into the FlexRay memory area. In this case, the CC clears the SDVEN or SIDEN bits immediately. If this trigger is set to 0 while SDVEN or SIDEN is set to 1, the CC writes continuously the enabled Sync Frame Tables into the FlexRay memory area. 0 Write continuously pairs of enabled Sync Frame Tables into FlexRay memory area. 1 Write only one pair of enabled Sync Frame Tables into FlexRay memory area.
14 SDVEN	Sync Frame Deviation Table Enable. This bit controls the generation of the Sync Frame Deviation Tables. The application must set this bit to request the CC to write the Sync Frame Deviation Tables into the FlexRay memory area. **NOTE:** If SDVEN is set to 1, then SIDEN must also be set to 1. 0 Do not write Sync Frame Deviation Tables 1 Write Sync Frame Deviation Tables into FlexRay memory area
15 SIDEN	Sync Frame ID Table Enable. This bit controls the generation of the Sync Frame ID Tables. The application must set this bit to 1 to request the CC to write the Sync Frame ID Tables into the FlexRay memory area. 0 Do not write Sync Frame ID Tables 1 Write Sync Frame ID Tables into FlexRay memory area

47.7.35 Sync Frame ID Rejection Filter Register (FR_SFIDRFR)

Write: Normal Mode

This register defines the Sync Frame Rejection Filter ID. The application must update this register outside of the static segment. If the application updates this register in the static segment, it can appear that the CC accepts the sync frame in the current cycle.

Address: 0h base + 46h offset = 46h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								SYNFRID							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFIDRFR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 SYNFRID	Sync Frame Rejection ID. This field defines the frame ID of a frame that must not be used for clock synchronization. For details see Sync Frame Rejection Filtering .

47.7.36 Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR)

Write: POC:config

This register defines the sync frame acceptance filter value. For details on filtering, see Sync Frame Filtering .

Address: 0h base + 48h offset = 48h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								FVAL							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFIDAFVR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 FVAL	Filter Value. This field defines the value for the sync frame acceptance filtering.

47.7.37 Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR)

Write: POC:config

This register defines the sync frame acceptance filter mask. For details on filtering see Sync Frame Acceptance Filtering .

Address: 0h base + 4Ah offset = 4Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						FMSK									
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SFIDAFMR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 FMSK	Filter Mask. This field defines the mask for the sync frame acceptance filtering.

47.7.38 Network Management Vector Register (FR_NMVR*n*)

Each of these six registers holds one part of the Network Management Vector. The length of the Network Management Vector is configured in the Network Management Vector Length Register (FR_NMVLR). If FR_NMVLR is programmed with a value that is less than 12 bytes, the remaining bytes of the Network Management Vector Registers (FR_NMVR0-FR_NMVR5), which are not used for the Network Management Vector accumulating, will remain 0.

The NMVR provides accrued information over all received NMVs in the last communication cycle. All NMVs received in one cycle are ORed into the NMVR. The NMVR is updated at the end of the communication cycle.

Address: 0h base + 4Ch offset + (2d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	NMVP															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_NMVR*n* field descriptions

Field	Description
0–15 NMVP	Network Management Vector Part. The mapping between the Network Management Vector Registers (FR_NMVR0-FR_NMVR5) and the receive message buffer payload bytes in NMV[0:11] is depicted in the following table.

Table 47-9. Mapping of NMVRn to the Received Payload Bytes NMVn

NMVRn Register	NMVn Received Payload
FR_NMVR0[NMVP[15:8]]	NMV0
FR_NMVR0[NMVP[7:0]]	NMV1
FR_NMVR1[NMVP[15:8]]	NMV2
FR_NMVR1[NMVP[7:0]]	NMV3
...	
FR_NMVR5[NMVP[15:8]]	NMV10
FR_NMVR5[NMVP[7:0]]	NMV11

47.7.39 Network Management Vector Length Register (FR_NMVLR)

Write: POC:config

This register defines the length of the network management vector in bytes.

Address: 0h base + 58h offset = 58h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0												NMVL			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_NMVLR field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 NMVL	Network Management Vector Length. protocol related variable: gNetworkManagementVectorLength This field defines the length of the Network Management Vector in bytes. Legal values are between 0 and 12.

47.7.40 Timer Configuration and Control Register (FR_TICCR)

Write: T2_CFG: POC:config T2_REP, T1_REP, T1SP, T2SP, T1TR, T2TR: Normal Mode

This register is used to configure and control the two timers T1 and T2. For timer details, see Timer Support . The Timer T1 is an absolute timer. The Timer T2 can be configured as an absolute or relative timer.

NOTE

Both timers are deactivated immediately when the protocol enters a state different from POC:normal active or POC:normal passive.

Address: 0h base + 5Ah offset = 5Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T2_CFG	T2_REP	0	0	0	T2ST	0			T1_REP	0	0	0	T1ST
Write						T2SP	T2TR							T1SP	T1TR	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TICCR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 T2_CFG	Timer T2 Configuration. This bit configures the timebase mode of Timer T2. 0 T2 is absolute timer. 1 T2 is relative timer.
3 T2_REP	Timer T2 Repetitive Mode. This bit configures the repetition mode of Timer T2. 0 T2 is non repetitive 1 T2 is repetitive
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 T2SP	Timer T2 Stop. This trigger bit is used to stop timer T2. 0 no effect 1 stop timer T2
6 T2TR	Timer T2 Trigger. This trigger bit is used to start timer T2. 0 no effect 1 start timer T2
7 T2ST	Timer T2 State. This status bit provides the current state of timer T2. 0 timer T2 is idle 1 timer T2 is running
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 T1_REP	Timer T1 Repetitive Mode. This bit configures the repetition mode of timer T1. 0 T1 is non repetitive 1 T1 is repetitive

Table continues on the next page...

FR_TICCR field descriptions (continued)

Field	Description
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 T1SP	Timer T1 Stop. This trigger bit is used to stop timer T1. 0 no effect 1 stop timer T1
14 T1TR	Timer T1 Trigger. This trigger bit is used to start timer T1. 0 no effect 1 start timer T1
15 T1ST	Timer T1 State. This status bit provides the current state of timer T1. 0 timer T1 is idle 1 timer T1 is running

47.7.41 Timer 1 Cycle Set Register (FR_TI1CYSR)

Write: Anytime

This register defines the cycle filter value and the cycle filter mask for timer T1. For a detailed description of timer T1, refer to Absolute Timer T1 .

NOTE

If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

Address: 0h base + 5Ch offset = 5Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T1_CYC_VAL						0		T1_CYC_MSK					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI1CYSR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 T1_CYC_VAL	Timer T1 Cycle Filter Value. This field defines the cycle filter value for timer T1.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 T1_CYC_MSK	Timer T1 Cycle Filter Mask. This field defines the cycle filter mask for timer T1.

47.7.42 Timer 1 Macrotick Offset Register (FR_TI1MTOR)

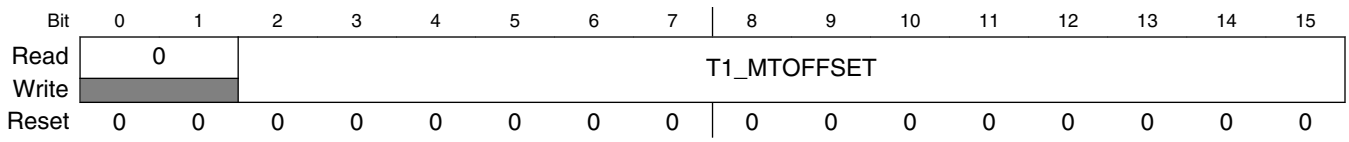
Write: Anytime

This register holds the macrotick offset value for timer T1. For a detailed description of timer T1, refer to Absolute Timer T1 .

NOTE

If the application modifies the value in this register while the timer is running, the change becomes effective immediately and timer T1 will expire according to the changed value.

Address: 0h base + 5Eh offset = 5Eh



FR_TI1MTOR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 T1_MTOFFSET	Timer 1 Macrotick Offset. This field defines the macrotick offset value for timer 1.

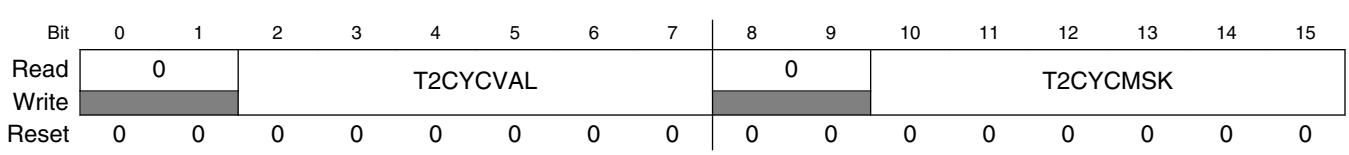
47.7.43 Timer 2 Configuration Register 0 (Absolute Timer Configuration) (FR_TI2CR0_ABS)

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, .

NOTE

If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and timer T2 will expire according to the changed values.

Address: 0h base + 60h offset = 60h



FR_TI2CR0_ABS field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 T2CYCVAL	Timer T2 Cycle Filter Mask
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 T2CYCMSK	Timer T2 Cycle Filter Mask

47.7.44 Timer 2 Configuration Register 0 (Relative Timer Configuration) (FR_TI2CR0_REL)

Write: Anytime

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to Relative Timer T2.

NOTE

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

Address: 0h base + 60h offset = 60h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	T2MTCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI2CR0_REL field descriptions

Field	Description
0–15 T2MTCNT	Timer T2 Macrotick High Word

47.7.45 Timer 2 Configuration Register 1 (Absolute Timer Configuration) (FR_TI2CR1_ABS)

Write: Anytime

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to Absolute Timer T2

NOTE

If timer T2 is configured as an absolute timer and the application modifies the values in this register while the timer is running, the change becomes effective immediately and the timer T2 will expire according to the changed values.

Address: 0h base + 62h offset = 62h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		T2MOFF													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI2CR1_ABS field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 T2MOFF	Timer T2 Macrotick Offset

47.7.46 Timer 2 Configuration Register 1 (Relative Timer Configuration) (FR_TI2CR1_REL)

Write: Anytime

The content of this register depends on the value of the T2_CFG bit in the Timer Configuration and Control Register (FR_TICCR). For a detailed description of timer T2, refer to Relative Timer T2.

NOTE

If timer T2 is configured as a relative timer and the application changes the values in this register while the timer is running, the change becomes effective when the timer has expired according to the old values.

Address: 0h base + 62h offset = 62h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	T2MTCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_TI2CR1_REL field descriptions

Field	Description
0–15 T2MTCNT	Timer T2 Macrotick Low Word

47.7.47 Slot Status Selection Register (FR_SSSR)

Write: Anytime

This register is used to access the four internal non-memory mapped slot status selection registers FR_SSSR0 to FR_SSSR3. Each internal register selects a slot, or symbol window/NIT, whose status vector will be saved in the corresponding Slot Status Registers (FR_SSR0-FR_SSR7) according to Slot Status Selection Register (FR_SSSR) . For a detailed description of slot status monitoring, refer to Slot Status Monitoring .

Table 47-10. Mapping Between FR_SSSRn and FR_SSRn

Internal Slot Status Selection Register	Write the Slot Status of the Slot Selected by FR_SSSRn for each			
	Even Communication Cycle		Odd Communication Cycle	
	For Channel B to	For Channel A to	For Channel B to	For Channel A to
FR_SSSR0	FR_SSR0[15:8]	FR_SSR0[7:0]	FR_SSR1[15:8]	FR_SSR1[7:0]
FR_SSSR1	FR_SSR2[15:8]	FR_SSR2[7:0]	FR_SSR3[15:8]	FR_SSR3[7:0]
FR_SSSR2	FR_SSR4[15:8]	FR_SSR4[7:0]	FR_SSR5[15:8]	FR_SSR5[7:0]
FR_SSSR3	FR_SSR6[15:8]	FR_SSR6[7:0]	FR_SSR7[15:8]	FR_SSR7[7:0]

Address: 0h base + 64h offset = 64h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0	SEL		0	SLOTNUMBER										
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSSR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_SSSR field descriptions (continued)

Field	Description
2–3 SEL	Selector. This field selects one of the four internal slot status selection registers for access. 00 select FR_SSSR0. 01 select FR_SSSR1. 10 select FR_SSSR2. 11 select FR_SSSR3.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 SLOTNUMBER	Slot Number. This field specifies the number of the slot whose status will be saved in the corresponding slot status registers. **NOTE:** If this value is set to 0, the related slot status register provides the status of the symbol window after the NIT start, and provides the status of the NIT after the cycle start.

47.7.48 Slot Status Counter Condition Register (FR_SSCCR)

Write: Anytime

This register is used to access and program the four internal non-memory mapped Slot Status Counter Condition Registers FR_SSCCR0 to FR_SSCCR3. Each of these four internal slot status counter condition registers defines the mode and the conditions for incrementing the counter in the corresponding Slot Status Counter Registers (FR_SSCR0-FR_SSCR3). The correspondence is given in Slot Status Counter Condition Register (FR_SSCCR) . For a detailed description of slot status counters, refer to Slot Status Counter Registers .

Table 47-11. Mapping between internal FR_SSCCRn and FR_SSCRn

Condition Register	Condition Defined for Register
FR_SSCCR0	FR_SSCR0
FR_SSCCR1	FR_SSCR1
FR_SSCCR2	FR_SSCR2
FR_SSCCR3	FR_SSCR3

Address: 0h base + 66h offset = 66h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	0	SEL		0	CNTCFG		MCY	VFR	SYF	NUF	SUF	STATUSMASK			
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSCCR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL field only on write access.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–3 SEL	Selector. This field selects one of the four internal slot counter condition registers for access. 00 select FR_SSCCR0. 01 select FR_SSCCR1. 10 select FR_SSCCR2. 11 select FR_SSCCR3.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–6 CNTCFG	Counter Configuration. These bit field controls the channel related incrementing of the slot status counter. 00 increment by 1 if condition is fulfilled on channel A. 01 increment by 1 if condition is fulfilled on channel B. 10 increment by 1 if condition is fulfilled on at least one channel. 11 increment by 2 if condition is fulfilled on both channels channel; increment by 1 if condition is fulfilled on only one channel.
7 MCY	Multi Cycle Selection. This bit defines whether the slot status counter accumulates over multiple communication cycles or provides information for the previous communication cycle only. 0 The Slot Status Counter provides information for the previous communication cycle only. 1 The Slot Status Counter accumulates over multiple communication cycles.
8 VFR	Valid Frame Restriction. This bit is used to restrict the counter to received valid frames. 0 The counter is not restricted to valid frames only. 1 The counter is restricted to valid frames only.
9 SYF	Sync Frame Restriction. This bit is used to restrict the counter to received frames with the sync frame indicator bit set to 1. 0 The counter is not restricted with respect to the sync frame indicator bit. 1 The counter is restricted to frames with the sync frame indicator bit set to 1.
10 NUF	Null Frame Restriction. This bit is used to restrict the counter to received frames with the null frame indicator bit set to 0. 0 The counter is not restricted with respect to the null frame indicator bit. 1 The counter is restricted to frames with the null frame indicator bit set to 0.
11 SUF	Startup Frame Restriction. This bit is used to restrict the counter to received frames with the startup frame indicator bit set to 1. 0 The counter is not restricted with respect to the startup frame indicator bit. 1 The counter is restricted to received frames with the startup frame indicator bit set to 1.
12–15 STATUSMASK	Slot Status Mask. This bit field is used to enable the counter with respect to the four slot status error indicator bits. STATUSMASK[3] - This bit enables the counting for slots with the syntax error indicator bit set to 1. STATUSMASK[2] - This bit enables the counting for slots with the content error indicator bit set to 1.

Table continues on the next page...

FR_SSCCR field descriptions (continued)

Field	Description
	STATUSMASK[1] - This bit enables the counting for slots with the boundary violation indicator bit set to 1. STATUSMASK[0] - This bit enables the counting for slots with the transmission conflict indicator bit set to 1.

47.7.49 Slot Status Register (FR_SSR*n*)

Each Slot Status Register (SSR) holds the status vector of the slot specified in the corresponding internal slot status selection register, which can be programmed using the Slot Status Selection Register (FR_SSSR). Each register is updated after the end of the corresponding slot as shown in Figure 47-31 . The register bits are directly related to the protocol variables and described in more detail in Slot Status Monitoring .

Address: 0h base + 68h offset + (2d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSR*n* field descriptions

Field	Description
0 VFB	Valid Frame on Channel B. protocol related variable: vSS!ValidFrame channel B 0 vSS!ValidFrame = 0 1 vSS!ValidFrame = 1
1 SYB	Sync Frame Indicator Channel B. protocol related variable: vRF!Header!SyFIndicator channel B 0 vRF!Header!SyFIndicator = 0 1 vRF!Header!SyFIndicator = 1
2 NFB	Null Frame Indicator Channel B. protocol related variable: vRF!Header!NFIndicator channel B 0 vRF!Header!NFIndicator = 0 1 vRF!Header!NFIndicator = 1
3 SUB	Startup Frame Indicator Channel B. protocol related variable: vRF!Header!SuFIndicator channel B 0 vRF!Header!SuFIndicator = 0 1 vRF!Header!SuFIndicator = 1
4 SEB	Syntax Error on Channel B. protocol related variable: vSS!SyntaxError channel B 0 vSS!SyntaxError = 0 1 vSS!SyntaxError = 1

Table continues on the next page...

FR_SSR*n* field descriptions (continued)

Field	Description
5 CEB	Content Error on Channel B. protocol related variable: vSS!ContentError channel B 0 vSS!ContentError = 0 1 vSS!ContentError = 1
6 BVB	Boundary Violation on Channel B. protocol related variable: vSS!BViolation channel B 0 vSS!BViolation = 0 1 vSS!BViolation = 1
7 TCB	Transmission Conflict on Channel B. protocol related variable: vSS!TxConflict channel B 0 vSS!TxConflict = 0 1 vSS!TxConflict = 1
8 VFA	Valid Frame on Channel A. protocol related variable: vSS!ValidFrame channel A 0 vSS!ValidFrame = 0 1 vSS!ValidFrame = 1
9 SYA	Sync Frame Indicator Channel A. protocol related variable: vRF!Header!SyFIndicator channel A 0 vRF!Header!SyFIndicator = 0 1 vRF!Header!SyFIndicator = 1
10 NFA	Null Frame Indicator Channel A. protocol related variable: vRF!Header!NFIndicator channel A 0 vRF!Header!NFIndicator = 0 1 vRF!Header!NFIndicator = 1
11 SUA	Startup Frame Indicator Channel A. protocol related variable: vRF!Header!SuFIndicator channel A 0 vRF!Header!SuFIndicator = 0 1 vRF!Header!SuFIndicator = 1
12 SEA	Syntax Error on Channel A. protocol related variable: vSS!SyntaxError channel A 0 vSS!SyntaxError = 0 1 vSS!SyntaxError = 1
13 CEA	Content Error on Channel A. protocol related variable: vSS!ContentError channel A 0 vSS!ContentError = 0 1 vSS!ContentError = 1
14 BVA	Boundary Violation on Channel A. protocol related variable: vSS!BViolation channel A 0 vSS!BViolation = 0 1 vSS!BViolation = 1
15 TCA	Transmission Conflict on Channel A. protocol related variable: vSS!TxConflict channel A 0 vSS!TxConflict = 0 1 vSS!TxConflict = 1

47.7.50 Slot Status Counter Register (FR_SSCR*n*)

Each of these four registers provides the slot status counter value for the previous communication cycle(s) and is updated at the cycle start. The provided value depends on the control bits and fields in the related internal slot status counter condition register FR_SSCCRn, which can be programmed by using the Slot Status Counter Condition Register (FR_SSCCR). For more details, see Slot Status Counter Registers .

NOTE

If the counter has reached its maximum value 0xFFFF and is in the multicycle mode, i.e. FR_SSCCRn[MCY] = 1, the counter is not reset to 0x0000. The application can reset the counter by clearing the FR_SSCCRn[MCY] bit and waiting for the next cycle start, when the CC clears the counter. Subsequently, the counter can be set into the multicycle mode again.

NOTE

Address: 0h base + 78h offset + (2d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	SLOTSTATUSCNT															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_SSCR*n* field descriptions

Field	Description
0–15 SLOTSTATUSCNT	Slot Status Counter. This field provides the current value of the Slot Status Counter.

47.7.51 MTS A Configuration Register (FR_MTSACFR)

Write: MTE: Anytime; CYCCNTMSK,CYCCNTVAL: POC:config

This register controls the transmission of the Media Access Test Symbol MTS on channel A. For more details, see MTS Generation .

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	MTE	0	CYCCNTMSK						0		CYCCNTVAL					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MTSACFR field descriptions

Field	Description
0 MTE	Media Access Test Symbol Transmission Enable. This control bit is used to enable and disable the transmission of the Media Access Test Symbol in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 CYCCNTMSK	Cycle Counter Mask. This field provides the filter mask for the MTS cycle count filter.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 CYCCNTVAL	Cycle Counter Value. This field provides the filter value for the MTS cycle count filter.

47.7.52 MTS B Configuration Register (FR_MTSBCFR)

Write: MTE: Anytime; CYCCNTMSK,CYCCNTVAL: POC:config

This register controls the transmission of the Media Access Test Symbol MTS on channel B. For more details, see MTS Generation .

Address: 0h base + 82h offset = 82h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	MTE	0	CYCCNTMSK						0		CYCCNTVAL					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_MTSBCFR field descriptions

Field	Description
0 MTE	Media Access Test Symbol Transmission Enable. This control bit is used to enable and disable the transmission of the Media Access Test Symbol in the selected set of cycles. 0 MTS transmission disabled 1 MTS transmission enabled
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–7 CYCCNTMSK	Cycle Counter Mask. This field provides the filter mask for the MTS cycle count filter.
8–9 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
10–15 CYCCNTVAL	Cycle Counter Value. This field provides the filter value for the MTS cycle count filter.

47.7.53 Receive Shadow Buffer Index Register (FR_RSBIR)

Write: WMD, SEL: Any Time; RSBIDX: POC:config

This register is used to provide and retrieve the indices of the message buffer header fields currently associated with the receive shadow buffers . For more details on the receive shadow buffer concept, refer to Receive Shadow Buffers Concept .

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7
Read Write	WMD	Reserved	SEL		Reserved			
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	RSBIDX							
Reset	0	0	0	0	0	0	0	0

FR_RSBIR field descriptions

Field	Description
0 WMD	Write Mode. This bit controls the write mode for this register. 0 update SEL and RSBIDX field on register write 1 update only SEL field on register write
1 Reserved	This field is reserved.
2–3 SEL	Selector. This field is used to select the internal receive shadow buffer index register for access. 00 FR_RSBIR_A1. receive shadow buffer index register for channel A, segment 1 01 FR_RSBIR_A2. receive shadow buffer index register for channel A, segment 2 10 FR_RSBIR_B1. receive shadow buffer index register for channel B, segment 1 11 FR_RSBIR_B2. receive shadow buffer index register for channel B, segment 2
4–7 Reserved	This field is reserved.
8–15 RSBIDX	RSBIDXA1/RSBIDXA2/RSBIDXB1/RSBIDXB2- Receive Shadow Buffer Index Receive Shadow Buffer Index. This field contains the current index of the message buffer header field of the receive shadow message buffer selected by the SEL field. The CC uses this index to determine the physical location of the shadow buffer header field in the FlexRay memory area. The CC will update this field during receive operation.The application provides initial message buffer header index value in the configuration phase. CC: Updates the message buffer header index after successful reception. Application: Provides initial message buffer header index. Legal Values are 0 <= i <= 131 . Illegal values will be detected during the message buffer search.

47.7.54 Receive FIFO Watermark and Selection Register (FR_RFWMSR)

Write: WMA/WMB: POC:config, SEL: Anytime

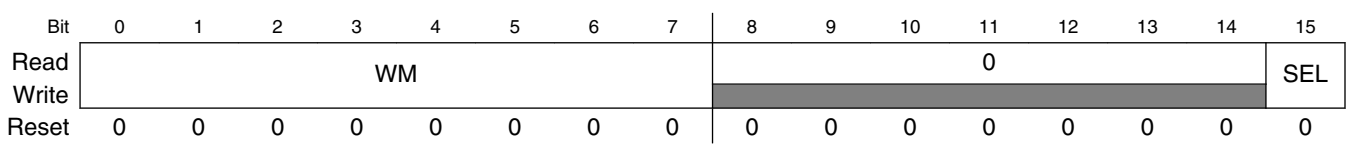
This register is used to

- select a receiver FIFO for subsequent programming access through the receiver FIFO configuration registers summarized in Receive FIFO Watermark and Selection Register (FR_RFWMSR) .
- to define the watermark for the selected FIFO.

Table 47-12. SEL Controlled Receiver FIFO Registers

Register
Receive FIFO Start Index Register (FR_RFSIR)
Receive FIFO Depth and Size Register (FR_RFDSR)
Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)
Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)
Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)
Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)
Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)
Receive FIFO Range Filter Control Register (FR_RFRFCTR)

Address: 0h base + 86h offset = 86h



FR_RFWMSR field descriptions

Field	Description
0–7 WM	WMA/WMB - Watermark This field defines the watermark value for the selected FIFO. This value is used to control the generation of the almost full interrupt flags.
8–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
15 SEL	Select. This control bit selects the receiver FIFO for subsequent programming. 0 Receiver FIFO for channel A selected 1 Receiver FIFO for channel B selected

47.7.55 Receive FIFO Start Index Register (FR_RFSIR)

Write: POC:config

This register defines the message buffer header index of the first message buffer of the selected FIFO.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								SIDX							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFSIR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 SIDX	SIDXA/SIDXB - Start Index This field defines the number of the message buffer header field of the first message buffer of the selected FIFO. The CC uses the value of the SIDX field to determine the physical location of the receiver FIFO's first message buffer header field.

47.7.56 Receive FIFO Depth and Size Register (FR_RFDSR)

Write: POC:config

This register defines the structure of the selected FIFO, i.e. the number of entries and the size of each entry .

Address: 0h base + 8Ah offset = 8Ah

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	FIFO_DEPTH								0	ENTRY_SIZE						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFDSR field descriptions

Field	Description
0–7 FIFO_DEPTH	FIFO_DEPTHA/FIFO_DEPTHB - FIFO Depth FIFO Depth. This field defines the depth of the selected FIFO, i.e. the number of entries. Note: If the FIFO_DEPTH is configured to 0, FR_RFFIDRFMR[FIDRFMSK] must be configured to 0 too, to ensure that no frames are received into the FIFO.

Table continues on the next page...

FR_RFDSR field descriptions (continued)

Field	Description
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9–15 ENTRY_SIZE	ENTRY_SIZEA/ENTRY_SIZEB - Entry Size This field defines the size of the frame data sections for the selected FIFO in 2 byte entities.

47.7.57 Receive FIFO A Read Index Register (FR_RFARIR)

This register provides the message buffer header index of the next available FIFO A entry that the application can read.

NOTE

If the FIFO is empty, the RDIDX field points to an physical message buffer with invalid content.

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0						RDIDX									
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFARIR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 RDIDX	Read Index. This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

47.7.58 Receive FIFO B Read Index Register (FR_RFBRIR)

This register provides the message buffer header index of the next available FIFO B entry that the application can read.

NOTE

If the FIFO is empty, the RDIDX field points to an physical message buffer with invalid content.

Address: 0h base + 8Eh offset = 8Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								RDIDX							
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFBRIR field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–15 RDIDX	Read Index. This field provides the message buffer header index of the next available FIFO message buffer that the application can read.

47.7.59 Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)

Write: POC:config

This register defines the filter value for the message ID acceptance filter of the selected FIFO. For details on message ID filtering see FIFO Filtering .

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MIDAFVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFMIDAFVR field descriptions

Field	Description
0–15 MIDAFVAL	MIDAFVALA/MIDAFVALB - Message ID Acceptance Filter Value Filter value for the message ID acceptance filter.

47.7.60 Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)

Write: POC:config

This register defines the filter mask for the message ID acceptance filter of the selected FIFO. For details on message ID filtering see FIFO Filtering .

Address: 0h base + 92h offset = 92h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MIDAFMSK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFMIDAFMR field descriptions

Field	Description
0–15 MIDAFMSK	MIDAFMSKA/MIDAFMSKB - Message ID Acceptance Filter Mask Filter mask for the message ID acceptance filter.

47.7.61 Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)

Write: POC:config

This register defines the filter value for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering see FIFO Filtering .

Address: 0h base + 94h offset = 94h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					FIDRFVAL										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFFIDRFVR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 FIDRFVAL	FIDRFVALA/FIDRFVALB - Frame ID Rejection Filter Value Filter value for the frame ID rejection filter.

47.7.62 Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)

Write: POC:config

This register defines the filter mask for the frame ID rejection filter of the selected FIFO. For details on frame ID filtering see FIFO Filtering .

Address: 0h base + 96h offset = 96h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					FIDRFMSK										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFFIDRFMR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 FIDRFMSK	Frame ID Rejection Filter Mask. Filter mask for the frame ID rejection filter.

47.7.63 Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)

Write: WMD, IBD, SEL: Any Time; SID: POC:config

This register provides access to the four internal frame ID range filter boundary registers of the selected FIFO. For details on frame ID range filter see FIFO Filtering .

Address: 0h base + 98h offset = 98h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0	IBD	SEL		0	SID										
Write	WMD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFRFCFR field descriptions

Field	Description
0 WMD	Write Mode. This control bit defines the write mode of this register. 0 Write to all fields in this register on write access. 1 Write to SEL and IBD field only on write access.
1 IBD	Interval Boundary. This control bit selects the interval boundary to be programmed with the SID value. 0 program lower interval boundary 1 program upper interval boundary
2–3 SEL	Filter Selector. This control field selects the frame ID range filter to be accessed. 00 select frame ID range filter 0. 01 select frame ID range filter 1. 10 select frame ID range filter 2. 11 select frame ID range filter 3.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_RFRFCFR field descriptions (continued)

Field	Description
5–15 SID	Slot ID Defines the IBD-selected frame ID boundary value for the SEL-selected range filter.

47.7.64 Receive FIFO Range Filter Control Register (FR_RFRFCTR)

Write: Anytime

This register is used to enable and disable each frame ID range filter and to define whether it is running as acceptance or rejection filter.

Address: 0h base + 9Ah offset = 9Ah

Bit	0	1	2	3	4	5	6	7
Read	0				F3MD	F2MD	F1MD	F0MD
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0				F3EN	F2EN	F1EN	F0EN
Write								
Reset	0	0	0	0	0	0	0	0

FR_RFRFCTR field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 F3MD	Range Filter 3 Mode. This control bit defines the filter mode of the frame ID range filter 3. 0 range filter 3 runs as acceptance filter 1 range filter 3 runs as rejection filter
5 F2MD	Range Filter 2 Mode. This control bit defines the filter mode of the frame ID range filter 2. 0 range filter 2 runs as acceptance filter 1 range filter 2 runs as rejection filter
6 F1MD	Range Filter 1 Mode. This control bit defines the filter mode of the frame ID range filter 1. 0 range filter 1 runs as acceptance filter 1 range filter 1 runs as rejection filter
7 F0MD	Range Filter 0 Mode. This control bit defines the filter mode of the frame ID range filter 0. 0 range filter 0 runs as acceptance filter 1 range filter 0 runs as rejection filter
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_RFRFCTR field descriptions (continued)

Field	Description
12 F3EN	Range Filter 3 Enable. This control bit is used to enable and disable the frame ID range filter 3. 0 range filter 3 disabled 1 range filter 3 enabled
13 F2EN	Range Filter 2 Enable. This control bit is used to enable and disable the frame ID range filter 2. 0 range filter 2 disabled 1 range filter 2 enabled
14 F1EN	Range Filter 1 Enable. This control bit is used to enable and disable the frame ID range filter 1. 0 range filter 1 disabled 1 range filter 1 enabled
15 F0EN	Range Filter 0 Enable. This control bit is used to enable and disable the frame ID range filter 0. 0 range filter 0 disabled 1 range filter 0 enabled

47.7.65 Last Dynamic Transmit Slot Channel A Register (FR_LDTXSLAR)

This register provides the number of the last transmission slot in the dynamic segment for channel A. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Address: 0h base + 9Ch offset = 9Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					LDYNTXSLOTA										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_LDTXSLAR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 LDYNTXSLOTA	Last Dynamic Transmission Slot Channel A. protocol related variable: zLastDynTxSlot channel A Number of the last transmission slot in the dynamic segment for channel A. If no frame was transmitted during the dynamic segment on channel A, the value of this field is set to 0.

47.7.66 Last Dynamic Transmit Slot Channel B Register (FR_LDTXSLBR)

This register provides the number of the last transmission slot in the dynamic segment for channel B. This register is updated after the end of the dynamic segment and before the start of the next communication cycle.

Address: 0h base + 9Eh offset = 9Eh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					LDYNTXSLOTB										
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_LDTXSLBR field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 LDYNTXSLOTB	Last Dynamic Transmission Slot Channel B. protocol related variable: zLastDynTxSlot channel B Number of the last transmission slot in the dynamic segment for channel B. If no frame was transmitted during the dynamic segment on channel B the value of this field is set to 0.

47.7.67 Protocol Configuration Register 0 (FR_PCR0)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Table 47-13. Protocol Configuration Register (PCR0-30) Fields

Name	Description	Min	Max	Unit	FR_PCR
coldstart_attempts	gColdstartAttempts			number	3
action_point_offset	gdActionPointOffset - 1			MT	0
cas_rx_low_max	gdCASRxLowMax - 1			gdBit	4
dynamic_slot_idle_phase	gdDynamicSlotIdlePhase			minislot	28
minislot_action_point_offset	gdMinislotActionPointOffset - 1			MT	3
minislot_after_action_point	gdMinislot - gdMinislotActionPointOffset - 1			MT	2
static_slot_length	gdStaticSlot			MT	0

Table continues on the next page...

Table 47-13. Protocol Configuration Register (PCR0-30) Fields (continued)

Name	Description	Min	Max	Unit	FR_PCR
static_slot_after_action_point	gdStaticSlot - gdActionPointOffset - 1			MT	13
symbol_window_exists	gdSymbolWindow!=0	0	1	bool	9
symbol_window_after_action_point	gdSymbolWindow - gdActionPointOffset - 1			MT	6
tss_transmitter	gdTSSTransmitter			gdBit	5
wakeup_symbol_rx_idle	gdWakeupSymbolRxIdle			gdBit	5
wakeup_symbol_rx_low	gdWakeupSymbolRxLow			gdBit	3
wakeup_symbol_rx_window	gdWakeupSymbolRxWindow			gdBit	4
wakeup_symbol_tx_idle	gdWakeupSymbolTxIdle			gdBit	8
wakeup_symbol_tx_low	gdWakeupSymbolTxLow			gdBit	5
noise_listen_timeout	(gListenNoise * pdListenTimeout) - 1			µT	16/17
macro_initial_offset_a	pMacroInitialOffset[A]			MT	6
macro_initial_offset_b	pMacroInitialOffset[B]			MT	16
macro_per_cycle	gMacroPerCycle			MT	10
macro_after_first_static_slot	gMacroPerCycle - gdStaticSlot			MT	1
macro_after_offset_correction	gMacroPerCycle - gOffsetCorrectionStart			MT	28
max_without_clock_correction_fatal	gMaxWithoutClockCorrectionFatal			cyclepairs	8
max_without_clock_correction_passive	gMaxWithoutClockCorrectionPassive			cyclepairs	8
minislot_exists	gNumberOfMinislots!=0	0	1	bool	9
minislots_max	gNumberOfMinislots - 1			minislot	29
number_of_static_slots	gNumberOfStaticSlots			static slot	2
offset_correction_start	gOffsetCorrectionStart			MT	11
payload_length_static	gPayloadLengthStatic			2 bytes	19
max_payload_length_dynamic	pPayloadLengthDynMax			2 bytes	24
first_minislot_action_point_offset	max(gdActionPointOffset, gdMinislotActionPointOffset) - 1			MT	13
allow_halt_due_to_clock	pAllowHaltDueToClock			bool	26
allow_passive_to_active	pAllowPassiveToActive			cyclepairs	12
cluster_drift_damping	pClusterDriftDamping			µT	24
comp_accepted_startup_range_a	pdAcceptedStartupRange - pDelayCompensation[A]			µT	22
comp_accepted_startup_range_b	pdAcceptedStartupRange - pDelayCompensation[B]			µT	26
listen_timeout	pdListenTimeout - 1			µT	14/15
key_slot_id	pKeySlotId			number	18
key_slot_used_for_startup	pKeySlotUsedForStartup			bool	11
key_slot_used_for_sync	pKeySlotUsedForSync			bool	11
latest_tx	gNumberOfMinislots - pLatestTx			minislot	21
sync_node_max	gSyncNodeMax			number	30
micro_initial_offset_a	pMicroInitialOffset[A]			µT	20

Table continues on the next page...

Table 47-13. Protocol Configuration Register (PCR0-30) Fields (continued)

Name	Description	Min	Max	Unit	FR_PCR
micro_initial_offset_b	pMicroInitialOffset[B]			µT	20
micro_per_cycle	pMicroPerCycle			µT	22/23
micro_per_cycle_min	pMicroPerCycle - pdMaxDrift			µT	24/25
micro_per_cycle_max	pMicroPerCycle + pdMaxDrift			µT	26/27
micro_per_macro_nom_half	round(pMicroPerMacroNom / 2)			µT	7
offset_correction_out	pOffsetCorrectionOut			µT	9
rate_correction_out	pRateCorrectionOut			µT	14
single_slot_enabled	pSingleSlotEnabled			bool	10
wakeup_channel	pWakeupChannel	see Protocol Configuration Register 0 (FR_PCR0)			10
wakeup_pattern	pWakeupPattern			number	18
decoding_correction_a	pDecodingCorrection + pDelayCompensation[A] + 2			µT	19
decoding_correction_b	pDecodingCorrection + pDelayCompensation[B] + 2			µT	7
key_slot_header_crc	header CRC for key slot	0x000	0x7FF	number	12
extern_offset_correction	pExternOffsetCorrection			µT	29
extern_rate_correction	pExternRateCorrection			µT	21

Table 47-14. Wakeup Channel Selection

wakeup_channel	Wakeup Channel
0	A
1	B

Address: 0h base + A0h offset = A0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	action_point_offset						static_slot_length									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR0 field descriptions

Field	Description
0–5 action_point_offset	gdActionPointOffset - 1
6–15 static_slot_length	gdStaticSlot

47.7.68 Protocol Configuration Register 1 (FR_PCR1)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A2h offset = A2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		macro_after_first_static_slot													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR1 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 macro_after_ first_static_slot	gMacroPerCycle - gdStaticSlot

47.7.69 Protocol Configuration Register 2 (FR_PCR2)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A4h offset = A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	minislot_after_action_point						number_of_static_slots									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR2 field descriptions

Field	Description
0–5 minislot_after_ action_point	gdMinislot - gdMinislotActionPointOffset - 1
6–15 number_of_ static_slots	gNumberOfStaticSlots

47.7.70 Protocol Configuration Register 3 (FR_PCR3)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A6h offset = A6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	wakeup_symbol_rx_low						minislot_action_point_offset					coldstart_attempts				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR3 field descriptions

Field	Description
0–5 wakeup_symbol_rx_low	gdWakeupSymbolRxLow
6–10 minislot_action_point_offset	gdMinislotActionPointOffset - 1
11–15 coldstart_attempts	gColdstartAttempts

47.7.71 Protocol Configuration Register 4 (FR_PCR4)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + A8h offset = A8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	cas_rx_low_max							wakeup_symbol_rx_window								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR4 field descriptions

Field	Description
0–6 cas_rx_low_max	gdCASRxLowMax - 1

Table continues on the next page...

FR_PCR4 field descriptions (continued)

Field	Description
7–15 wakeup_symbol_ rx_window	gdWakeupSymbolRxWindow

47.7.72 Protocol Configuration Register 5 (FR_PCR5)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + AAh offset = AAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	tss_transmitter				wakeup_symbol_tx_low						wakeup_symbol_rx_idle					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR5 field descriptions

Field	Description
0–3 tss_transmitter	gdTSSTransmitter
4–9 wakeup_symbol_ tx_low	gdWakeupSymbolTxLow
10–15 wakeup_symbol_ rx_idle	gdWakeupSymbolRxIdle

47.7.73 Protocol Configuration Register 6 (FR_PCR6)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + ACh offset = ACh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	0	symbol_window_after_action_point							macro_initial_offset_a							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR6 field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–8 symbol_window_after_action_point	gdSymbolWindow - gdActionPointOffset - 1
9–15 macro_initial_offset_a	pMacroInitialOffset[A]

47.7.74 Protocol Configuration Register 7 (FR_PCR7)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + AEh offset = AEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	decoding_correction_b								micro_per_macro_nom_half							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR7 field descriptions

Field	Description
0–8 decoding_correction_b	pDecodingCorrection + pDelayCompensation[B] + 2
9–15 micro_per_macro_nom_half	round(pMicroPerMacroNom / 2)

47.7.75 Protocol Configuration Register 8 (FR_PCR8)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B0h offset = B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	max_without_clock_correction_fatal				max_without_clock_correction_passive				wakeup_symbol_tx_idle							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR8 field descriptions

Field	Description
0–3 max_without_clock_correction_fatal	gMaxWithoutClockCorrectionFatal
4–7 max_without_clock_correction_passive	gMaxWithoutClockCorrectionPassive
8–15 wakeup_symbol_tx_idle	gdWakeupSymbolTxIdle

47.7.76 Protocol Configuration Register 9 (FR_PCR9)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B2h offset = B2h

Bit	0	1	2	3	4	5	6	7
Read Write	minislot_exists	symbol_window_exists	offset_correction_out					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	offset_correction_out							
Reset	0	0	0	0	0	0	0	0

FR_PCR9 field descriptions

Field	Description
0 minislot_exists	gNumberOfMinislots!=0
1 symbol_window_exists	gdSymbolWindow!=0

Table continues on the next page...

FR_PCR9 field descriptions (continued)

Field	Description
2–15 offset_ correction_out	pOffsetCorrectionOut

47.7.77 Protocol Configuration Register 10 (FR_PCR10)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7
Read Write	single_slot_ enabled	wakeup_ channel	macro_per_cycle					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	macro_per_cycle							
Reset	0	0	0	0	0	0	0	0

FR_PCR10 field descriptions

Field	Description
0 single_slot_ enabled	pSingleSlotEnabled
1 wakeup_channel	pWakeupChannel
2–15 macro_per_cycle	gMacroPerCycle

47.7.78 Protocol Configuration Register 11 (FR_PCR11)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B6h offset = B6h

Bit	0	1	2	3	4	5	6	7
Read Write	key_slot_used_for_startup	key_slot_used_for_sync	offset_correction_start					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	offset_correction_start							
Reset	0	0	0	0	0	0	0	0

FR_PCR11 field descriptions

Field	Description
0 key_slot_used_for_startup	pKeySlotUsedForStartup
1 key_slot_used_for_sync	pKeySlotUsedForSync
2–15 offset_correction_start	gOffsetCorrectionStart

47.7.79 Protocol Configuration Register 12 (FR_PCR12)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	allow_passive_to_active					key_slot_header_crc										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR12 field descriptions

Field	Description
0–4 allow_passive_to_active	pAllowPassiveToActive
5–15 key_slot_header_crc	header CRC for key slot

47.7.80 Protocol Configuration Register 13 (FR_PCR13)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + BAh offset = BAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	first_minislot_action_point_offset						static_slot_after_action_point									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR13 field descriptions

Field	Description
0–5 first_minislot_action_point_offset	max(gdActionPointOffset, gdMinislotActionPointOffset) - 1
6–15 static_slot_after_action_point	gdStaticSlot - gdActionPointOffset - 1

47.7.81 Protocol Configuration Register 14 (FR_PCR14)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	rate_correction_out											listen_timeout				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR14 field descriptions

Field	Description
0–10 rate_correction_out	pRateCorrectionOut
11–15 listen_timeout	pdListenTimeout - 1

47.7.82 Protocol Configuration Register 15 (FR_PCR15)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + BEh offset = BEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	listen_timeout															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR15 field descriptions

Field	Description
0–15 listen_timeout	pdListenTimeout - 1

47.7.83 Protocol Configuration Register 16 (FR_PCR16)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C0h offset = C0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	macro_initial_offset_b							noise_listen_timeout								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR16 field descriptions

Field	Description
0–6 macro_initial_ offset_b	pMacroInitialOffset[B]
7–15 noise_listen_ timeout	(gListenNoise * pdListenTimeout) - 1

47.7.84 Protocol Configuration Register 17 (FR_PCR17)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C2h offset = C2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	noise_listen_timeout															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR17 field descriptions

Field	Description
0–15 noise_listen_ timeout	(gListenNoise * pdListenTimeout) - 1

47.7.85 Protocol Configuration Register 18 (FR_PCR18)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C4h offset = C4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	wakeup_pattern						key_slot_id									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR18 field descriptions

Field	Description
0–5 wakeup_pattern	pWakeupPattern
6–15 key_slot_id	pKeySlotId

47.7.86 Protocol Configuration Register 19 (FR_PCR19)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C6h offset = C6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	decoding_correction_a								payload_length_static							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR19 field descriptions

Field	Description
0–8 decoding_ correction_a	pDecodingCorrection + pDelayCompensation[A] + 2
9–15 payload_length_ static	gPayloadLengthStatic

47.7.87 Protocol Configuration Register 20 (FR_PCR20)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + C8h offset = C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_initial_offset_b								micro_initial_offset_a							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR20 field descriptions

Field	Description
0–7 micro_initial_ offset_b	pMicroInitialOffset[A]
8–15 micro_initial_ offset_a	pMicroInitialOffset[A]

47.7.88 Protocol Configuration Register 21 (FR_PCR21)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + CAh offset = CAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	extern_rate_ correction			latest_tx												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR21 field descriptions

Field	Description
0–2 extern_rate_ correction	pExternRateCorrection
3–15 latest_tx	gNumberOfMinislots - pLatestTx

47.7.89 Protocol Configuration Register 22 (FR_PCR22)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + CCh offset = CCh

Bit	0	1	2	3	4	5	6	7
Read Write	Reserved	comp_accepted_startup_range_a						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	comp_accepted_startup_range_a				micro_per_cycle			
Reset	0	0	0	0	0	0	0	0

FR_PCR22 field descriptions

Field	Description
0 Reserved	Reserved bit, will not be changed. Application must not write any value different from the reset value.

Table continues on the next page...

FR_PCR22 field descriptions (continued)

Field	Description
	This field is reserved.
1–11 comp_accepted_ startup_range_a	pdAcceptedStartupRange - pDelayCompensation[A]
12–15 micro_per_cycle	pMicroPerCycle

47.7.90 Protocol Configuration Register 23 (FR_PCR23)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + CEh offset = CEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_per_cycle															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR23 field descriptions

Field	Description
0–15 micro_per_cycle	pMicroPerCycle

47.7.91 Protocol Configuration Register 24 (FR_PCR24)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	cluster_drift_damping					max_payload_length_dynamic							micro_per_cycle_min			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR24 field descriptions

Field	Description
0–4 cluster_drift_ damping	pClusterDriftDamping
5–11 max_payload_ length_dynamic	pPayloadLengthDynMax
12–15 micro_per_cycle_ min	pMicroPerCycle - pdMaxDrift

47.7.92 Protocol Configuration Register 25 (FR_PCR25)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D2h offset = D2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_per_cycle_min															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR25 field descriptions

Field	Description
0–15 micro_per_cycle_ min	pMicroPerCycle - pdMaxDrift

47.7.93 Protocol Configuration Register 26 (FR_PCR26)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D4h offset = D4h

Bit	0	1	2	3	4	5	6	7
Read Write	allow_halt_due_to_clock	comp_accepted_startup_range_b						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	comp_accepted_startup_range_b				micro_per_cycle_max			
Reset	0	0	0	0	0	0	0	0

FR_PCR26 field descriptions

Field	Description
0 allow_halt_due_to_clock	pAllowHaltDueToClock
1–11 comp_accepted_startup_range_b	pdAcceptedStartupRange - pDelayCompensation[B]
12–15 micro_per_cycle_max	pMicroPerCycle + pdMaxDrift

47.7.94 Protocol Configuration Register 27 (FR_PCR27)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D6h offset = D6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	micro_per_cycle_max															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR27 field descriptions

Field	Description
0–15 micro_per_cycle_max	pMicroPerCycle + pdMaxDrift

47.7.95 Protocol Configuration Register 28 (FR_PCR28)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + D8h offset = D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	dynamic_slot_idle_phase		macro_after_offset_correction													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR28 field descriptions

Field	Description
0–1 dynamic_slot_idle_phase	gdDynamicSlotIdlePhase.
2–15 macro_after_offset_correction	gMacroPerCycle - gOffsetCorrectionStart

47.7.96 Protocol Configuration Register 29 (FR_PCR29)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + DAh offset = DAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	extern_offset_correction			minislots_max												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR29 field descriptions

Field	Description
0–2 extern_offset_correction	pExternOffsetCorrection

Table continues on the next page...

FR_PCR29 field descriptions (continued)

Field	Description
3–15 minislots_max	gNumberOfMinislots - 1

47.7.97 Protocol Configuration Register 30 (FR_PCR30)

Write: POC:config

The protocol configuration registers (FR_PCRn) provide the necessary configuration information to the protocol engine. The individual values in the registers are described in Protocol Configuration Register 0 (FR_PCR0) .

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0												sync_node_max			
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_PCR30 field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 sync_node_max	gSyncNodeMax

47.7.98 StopWatch Count High Register (FR_STPWHR)

Address: 0h base + DEh offset = DEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	STPW_S_H															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_STPWHR field descriptions

Field	Description
0–15 STPW_S_H	StopWatch Count High These are the high 16 bits of the stopwatch counter running on system clock. See Stop Watch Count Low Register (STPWLR) for the low 16 bits. This counter starts on cycle start even and stops on an external event “OS_tick”. When all 32 bits of the stopwatch counter (in STPWHR and STPWLR) are read, the counter automatically clears. For details, please see StopWatch Function section.

47.7.99 Stop Watch Count Low Register (FR_STPWLR)

Address: 0h base + E0h offset = E0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	STPW_S_L															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_STPWLR field descriptions

Field	Description
0–15 STPW_S_L	StopWatch Count Low These are the low 16 bits of the stopwatch counter running on system clock. See Stop Watch Count High Register (STPWHR) for the high 16 bits. This counter starts on cycle start even and stops on an external event “OS_tick”. When all 32 bits of the stopwatch counter (in STPWHR and STPWLR) are read, the counter automatically clears. For details, please see StopWatch Function section.

47.7.100 Protocol Event Output Enable and StopWatch Control Register (FR_PEOER)

Write: Anytime

This register defines whether or not the Event output ports are enabled based on the events such as Cycle start and Timer1 and Timer2 expiry (see Timer Support for details).

Address: 0h base + E2h offset = E2h

Bit	0	1	2	3	4	5	6	7
Read	1							STPW_EN
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0					TIM2_EE	TIM1_EE	CYC_EE
Write								
Reset	0	0	0	0	0	0	0	0

FR_PEOER field descriptions

Field	Description
0–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
7 STPW_EN	Stopwatch count Enable This bit controls the stopwatch counter.

Table continues on the next page...

FR_PEOER field descriptions (continued)

Field	Description
	0 Stopwatch counter disabled 1 Stopwatch counter enabled
8–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 TIM2_EE	Timer 2 expired Event Output Enable- This bit controls the event on port "fr_evt_tim2" 0 Timer 2 expired event out disabled 1 Timer 2 expired event out enabled
14 TIM1_EE	Timer 1 expired Event Output Enable- This bit controls the event on port "fr_evt_tim1" 0 Timer 1 expired event out disabled 1 Timer 1 expired event out enabled
15 CYC_EE	Cycle Start Event Output Enable- This bit controls the event on port "fr_evt_cyc" 0 Cycle start event out disabled 1 Cycle start event out enabled

47.7.101 Receive FIFO Start Data Offset Register (FR_RFSDOR)

Write: POC:config

NOTE

Since all data fields of the FIFO are of equal length and are located at subsequent system memory addresses the content of the FR_RFSDOR corresponds to the start address of payload area of the selected FIFO.

Address: 0h base + E6h offset = E6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SDO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFSDOR field descriptions

Field	Description
0–15 SDO	SDOA/SDOB - Start Data Field Offset Start Data Field Offset. This field defines the data field offset of the header field of the first message buffer of the selected FIFO. The CC uses the value of the SDO field to determine the physical location of the receiver FIFO's first message buffer header field. For configuration constraints see Configure Data Field Offsets .

47.7.102 Receive FIFO System Memory Base Address High Register (FR_RFSYMBADHR)

These registers define the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. The system memory base address is used by the BMIF to calculate the physical memory address for system memory accesses for the FIFOs.

Write: Disabled Mode

Address: 0h base + E8h offset = E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SMBA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFSYMBADHR field descriptions

Field	Description
0–15 SMBA	System Memory Base Address. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.

47.7.103 Receive FIFO System Memory Base Address Low Register (FR_RFSYMBADLR)

These registers define the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. The system memory base address is used by the BMIF to calculate the physical memory address for system memory accesses for the FIFOs.

Write: Disabled Mode

Address: 0h base + EAh offset = EAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SMBA												0			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFSYMBADLR field descriptions

Field	Description
0–11 SMBA	System Memory Base Address. This is the value of the system memory base address for the receive FIFO if the FIFO address mode bit FR_MCR[FAM] is set to 1. It is defines as a byte address.
12–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

47.7.104 Receive FIFO Periodic Timer Register (FR_RFPTR)

Write: POC:config

This register holds periodic timer duration for the periodic FIFO timer. The periodic timer applies to both FIFOs (see FIFO Periodic Timer).

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0		PTD													
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFPTR field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 PTD	Periodic Timer Duration. This value defines the periodic timer duration in terms of macroticks. 0000 timer stays expired 3FFF timer never expires other timer expires after specified number of macroticks, expires and is restarted at each cycle start

47.7.105 Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)

This register provides the current fill level of the two receiver FIFOs and is used to pop a number of entries from the FIFOs

If the pop count value PCA/PCB is greater than the current FIFO fill level FLB/FLA, than the FIFO is empty after the update. No notification is given that not the required number of entries was removed.

.

Address: 0h base + EEh offset = EEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	FLB_or_PCB								FLA_or_PCA							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_RFFLPCR field descriptions

Field	Description
0–7 FLB_or_PCB	**NOTE:** The name and function of the fields in this register vary depending on whether they are being read or written. See the following table for details.

Operation	Sub-Field	Description
Read	0-7 FLB	Fill Level FIFO B. This field provides the current number of entries in the FIFO B.
	-	-
Write	0-7 PCB	Pop Count FIFO B. This field defines the number of entries to be removed from FIFO B.
	-	-

Field	Description
8–15 FLA_or_PCA	

Operation	Sub-Field	Description
Read	-	-
	8-15 FLA	Fill Level FIFO A- This field provides the current number of entries in the FIFO A.
Write	-	-
	8-15 PCA	Pop Count FIFO A - This field defines the number of entries to be removed from FIFO A.

47.7.106 ECC Error Interrupt Flag and Enable Register (FR_EEIFER)

This register provides the means to control the ECC related interrupt request lines and provides the corresponding interrupt flags. The interrupt flags are cleared by writing 1, which resets the corresponding report registers. For a detailed description see Memory Error Reporting .

Address: 0h base + F0h offset = F0h

Bit	0	1	2	3	4	5	6	7
Read	LRNE_OF	LRCE_OF	DRNE_OF	DRCE_OF	LRNE_IF	LRCE_IF	DRNE_IF	DRCE_IF
Write	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0				LRNE_IE	LRCE_IE	DRNE_IE	DRCE_IE
Write								
Reset	0	0	0	0	0	0	0	0

FR_EEIFER field descriptions

Field	Description
0 LRNE_OF	LRAM Non-Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears: a) memory errors are detected but not corrected on CHI LRAM and interrupt flag LRNE_IF is already 1. b) memory errors are detected but not corrected on at least two banks of CHI LRAM 0 no such event 1 Non-Corrected Error overflow detected on CHI LRAM
1 LRCE_OF	LRAM Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears a) memory errors are detected and corrected on CHI LRAM and interrupt flag LRCE_IF is already 1. b) memory errors are detected and corrected on at least two banks of CHI LRAM **NOTE:** Error Correction not implemented on CHI LRAM, flag will never be asserted. 0 no such event 1 Corrected Error overflow detected on CHI LRAM
2 DRNE_OF	DRAM Non-Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears: a) memory errors are detected but not corrected on PE DRAM and interrupt flag DRNE_IF is already 1. b) memory errors are detected but not corrected on at least two banks of the PE DRAM 0 no such event 1 Non-Corrected Error overflow detected on PE DRAM

Table continues on the next page...

FR_EEIFER field descriptions (continued)

Field	Description
3 DRCE_OF	DRAM Corrected Error Overflow Flag. This flag is set to 1 when at least one of the following events appears: a) memory errors are detected and corrected on PE DRAM and interrupt flag DRCE_IF is already 1. b) memory errors are detected and corrected on at least two banks of PE DRAM 0 no such event 1 Corrected Error overflow detected on PE DRAM
4 LRNE_IF	LRAM Non-Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected but not corrected on the CHI LRAM. 0 no such event 1 Non-Corrected Error detected on CHI LRAM
5 LRCE_IF	LRAM Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected and corrected on the CHI LRAM. **NOTE:** Error Correction not implemented on CHI LRAM, flag will never be asserted. 0 no such event 1 Corrected Error detected on CHI LRAM
6 DRNE_IF	DRAM Non-Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected but not corrected on PE DRAM. 0 no such event 1 Non-Corrected Error detected on PE DRAM
7 DRCE_IF	DRAM Corrected Error Interrupt Flag. This interrupt flag is set to 1 when a memory error is detected and corrected on PE DRAM. 0 no such event 1 Corrected Error detected on PE DRAM
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 LRNE_IE	LRAM Non-Corrected Error Interrupt Enable. This flag controls if the LRAM Non-Corrected Error Interrupt line is asserted when the LRNE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
13 LRCE_IE	LRAM Corrected Error Interrupt Enable. This flag controls if the LRAM Corrected Error Interrupt line is asserted when the LRCE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
14 DRNE_IE	DRAM Non-Corrected Error Interrupt Enable. This flag controls if the DRAM Non-Corrected Error Interrupt line is asserted when the DRNE_IF flag is set. 0 Disable interrupt line 1 Enable interrupt line
15 DRCE_IE	DRAM Corrected Error Interrupt Enable. This flag controls if the DRAM Corrected Error Interrupt line is asserted when the DRCE_IF flag is set.

Table continues on the next page...

FR_EEIFER field descriptions (continued)

Field	Description
	0 Disable interrupt line 1 Enable interrupt line

47.7.107 ECC Error Report and Injection Control Register (FR_EERICR)

This register configures the error injection and error reporting and provides the selector for the content of the report registers.

Address: 0h base + F2h offset = F2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	BSY	0					ERS		0			ERM	0		EIM	EIE
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERICR field descriptions

Field	Description
0 BSY	Register Update Busy- This field indicates the current state of the ECC configuration update and controls the register write access condition IDL specified in " Register Write Access 0 ECC configuration is idle 1 ECC configuration is running
1–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 ERS	Error Report Select. This field selects the content of the ECC Error reporting registers. 00 show PE DRAM non-corrected error information 01 show PE DRAM corrected error information 10 show CHI LRAM non-corrected error information 11 show CHI LRAM corrected error information
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 ERM	Error Report Mode. This bit configures the type of data written into the internal error report registers on the detection of a memory error. 0 store data and code as delivered by ecc decoding logic. 1 store data and code as read from the memory.
12–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 EIM	Error Injection Mode. This bit configures the ECC error injection mode.

Table continues on the next page...

FR_EERICR field descriptions (continued)

Field	Description
	0 use FR_EEIDR[DATA] and FR_EEICR[CODE] as XOR distortion pattern for error injection. 1 use FR_EEIDR[DATA] and FR_EEICR[CODE] as write value for error injection.
15 EIE	Error Injection Enable. This bit configures the ECC error injection on the memories. 0 Error injection disabled 1 Error injection enabled

47.7.108 ECC Error Report Address Register (FR_EERAR)

This register provides the memory identifier, bank, and address for which the memory error is reported.

Address: 0h base + F4h offset = F4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	MID	BANK			ADDR											
Write																
Reset	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERAR field descriptions

Field	Description
0 MID	Memory Identifier. This flag provides the memory instance for which the memory error is reported. 0 PE DRAM 1 CHI LRAM
1–3 BANK	Memory Bank. This field provides the BANK for which the memory error is reported. 111 reset value, indicates no error found after reset. For MID=0: 000 PE DRAM [7:0] 001 PE DRAM [15:8] others - not used For MID=1: Refer to the following table for the assignment of the LRAM banks.

Table 47-15. LRAM Bank Value for MID = 1

BANK	Register		
000	FR_MBCCFR(2n)	FR_MBDOR(6n)	FR_LEETR0
001	FR_MBFIDR(2n)	FR_MBDOR(6n + 1)	FR_LEETR1
010	FR_MBIDXR(2n)	FR_MBDOR(6n + 2)	FR_LEETR2
011	FR_MBCCFR(2n+1)	FR_MBDOR(6n + 3)	FR_LEETR3

Table continues on the next page...

FR_EERAR field descriptions (continued)

Field	Description
	Table 47-15. LRAM Bank Value for MID = 1 (continued)

BANK	Register		
100	FR_MBFIDR(2n+1)	FR_MBDOR(6n + 4)	FR_LEETR4
101	FR_MBIDXR(2n+1)	FR_MBDOR(6n + 5)	FR_LEETR5
110	Not Used	Not Used	Not Used
111			

4–15 ADDR	Memory Address. This field provides the address of the failing memory location.

47.7.109 ECC Error Report Data Register (FR_EERDR)

This register provides the data related information of the reported memory read access. The assignment of the bits depends on the selected memory and memory bank as shown in ECC Error Report Data Register (FR_EERDR) .

Table 47-16. Valid Bits in FR_EERDR[DATA] / FR_EEIDR[DATA] field

MEM	BANK	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PE DRAM	0									PE DRAM[7:0]							
PE DRAM	1									PE DRAM[15:8]							
CHI LRAM	0	FR_MBCCFR(2n)															
CHI LRAM	0	FR_MBDOR(6n)															
CHI LRAM	0	FR_LEETR0															
CHI LRAM	1						FR_MBFIDR(2n)[FID]										
CHI LRAM	1	FR_MBDOR(6n+1)															
CHI LRAM	1	FR_LEETR1															
CHI LRAM	2									FR_MBIDXR(2n)[MBIDX]							
CHI LRAM	2	FR_MBDOR(6n+2)															
CHI LRAM	3	FR_MBCCFR(2n+1)															
CHI LRAM	3	FR_MBDOR(6n+3)															
CHI LRAM	3	FR_LEETR3															
CHI LRAM	4						FR_MBFIDR(2n+1)[FID]										
CHI LRAM	4	FR_MBDOR(6n+4)															
CHI LRAM	4	FR_LEETR4															
CHI LRAM	5									FR_MBIDXR(2n+1)[MBIDX]							
CHI LRAM	5	FR_MBDOR(6n+5)															
CHI LRAM	5	FR_LEETR5															

Address: 0h base + F6h offset = F6h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	DATA															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERDR field descriptions

Field	Description
0–15 DATA	Data. The content of this field depends on the report mode selected by FR_EERICR[ERM] ERM=0: Ecc Data, shows data as generated by the ecc decoding logic. ERM=1: Memory Data, shows data as read from the memory.

47.7.110 ECC Error Report Code Register (FR_EERCR)

This register provides the ECC related information of the reported memory read access.

Address: 0h base + F8h offset = F8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0											CODE				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EERCR field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 CODE	Code. The content of this field depends on the report mode selected by FR_EERICR[ERM] ERM=0: Syndrome. Shows the ecc syndrome generated by the ecc decoding logic. The coding of the PE DRAM syndrome is shown in PE DRAM Syndrome . The coding of the CHI LRAM syndrome is shown in CHI LRAM Syndrome . ERM=1: Checkbits. Shows the ecc checkbits read from the memory.

47.7.111 ECC Error Injection Address Register (FR_EEIAR)

This register defines the memory module, bank, and address where the ECC error has to be injected.

Address: 0h base + FAh offset = FAh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MID	BANK			ADDR											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EEIAR field descriptions

Field	Description
0 MID	Memory Identifier. This flag defines the memory instance for ECC error injection. 0 PE DRAM 1 CHI LRAM
1–3 BANK	Memory Bank. This field defines the memory bank for ECC error injection. For MID=0: 000 BANK0: PE DRAM [7:0] 001 BANK1: PE DRAM [15:8] others reserved For MID=1: Refer to ECC Error Report Address Register (FR_EERAR) for the assignment of the LRAM banks.
4–15 ADDR	Memory Address. This flag defines the memory address for ECC error injection.

47.7.112 ECC Error Injection Data Register (FR_EEIDR)

This register defines the data distortion pattern for the error injection write. The number of valid bits depends on the selected memory and memory bank as shown in ECC Error Report Data Register (FR_EERDR) .

NOTE

The effect of the error injected depends from the LRAM content at the address accessed and from the module internal usage of the data. Refer to Memory Error Response for details.

Address: 0h base + FCh offset = FCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EEIDR field descriptions

Field	Description
0–15 DATA	Data. The content of this field depends on the error injection mode selected by FR_EERICR[EIM]. EIM=0: This field defines the XOR distortion pattern for the data written into the memory. EIM=1: This field defines the data to be written into the memory.

47.7.113 ECC Error Injection Code Register (FR_EEICR)

This register defines the ECC code distortion pattern for the error injection write.

Address: 0h base + FEh offset = FEh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0											CODE				
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FR_EEICR field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 CODE	Code. The content of this field depends on the error injection mode selected by FR_EERICR[EIM]. EIM=0: This field defines the XOR distortion pattern for the ecc checkbits written into the memory. EIM=1: This field defines the ecc checkbits written into the memory.

47.7.114 Message Buffer Configuration, Control, Status Register (FR_MBCCSR*n*)

Write: MTD: POC:config or MB_DIS CMT: MB_LCK or MB_DIS EDT, LCKT, MBIE, MBIF: Normal Mode

The content of these registers is composed of message buffer configuration data, message buffer control data, message buffer status information, and message buffer interrupt flags. A detailed description of all flags can be found in Individual Message Buffer Functional Description.

If the application writes 1 to the EDT bit, no write access to the other register bits is performed.

If the application writes 0 to the EDT bit and 1 to the LCKT bit, no write access to the other bits is performed.

Address: 0h base + 800h offset + (8d × i), where i=0d to 127d

Bit	0	1	2	3	4	5	6	7
Read	0			MTD	CMT		0	MBIE
Write						EDT	LCKT	
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0			DUP	DVAL	EDS	LCKS	MBIF
Write								w1c
Reset	0	0	0	0	0	0	0	0

FR_MBCCSR*n* field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 MTD	Message Buffer Transfer Direction. This bit configures the transfer direction of a message buffer. 0 Receive message buffer 1 Transmit message buffer
4 CMT	Commit for Transmission. This bit indicates if the transmit message buffer data are ready for transmission. **NOTE:** This bit is read/write but may be modified by hardware other than by a reset. 0 Message buffer data not ready for transmission 1 Message buffer data ready for transmission
5 EDT	Enable/Disable Trigger. If the application writes 1 to this bit, a message buffer enable or disable is triggered, depending on the current value of the EDS status bit. 0 No effect 1 Message buffer enable or disable is triggered
6 LCKT	Lock/Unlock Trigger. If the application writes 1 to this bit and writes 0 to the EDT bit, a message buffer lock or unlock is triggered, depending on the current value of the LCKS status bit. 0 No effect 1 Message buffer lock or unlock is triggered
7 MBIE	Message Buffer Interrupt Enable. This control bit defines whether the message buffer will generate an interrupt request when its MBIF flag is set. 0 Interrupt request generation disabled 1 Interrupt request generation enabled
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FR_MBCCSR*n* field descriptions (continued)

Field	Description
11 DUP	Data Updated. This status bit indicates whether the frame header in the message buffer header field and the data in the message buffer data field were updated after a frame reception. 0 Frame Header and Message buffer data field not updated 1 Frame Header and Message buffer data field updated
12 DVAL	Data Valid. For receive message buffers this status bit indicates whether the message buffer data field contains valid frame data. For transmit message buffers the status bit indicates if a message is transferred again due to the state transmission mode of the message buffer. 0 receive message buffer contains no valid frame data / message is transmitted for the first time 1 receive message buffer contains valid frame data / message will be transferred again
13 EDS	Enable/Disable Status. This status bit indicates whether the message buffer is enabled or disabled. 0 Message buffer is disabled. 1 Message buffer is enabled.
14 LCKS	Lock Status. This status bit indicates the current lock status of the message buffer. 0 Message buffer is not locked by the application. 1 Message buffer is locked by the application.
15 MBIF	Message Buffer Interrupt Flag. This flag is set when the slot status field of the message buffer was updated after frame transmission or reception, or when a transmit message buffer was just enabled by the application. 0 No such event 1 Slot status field updated or transmit message buffer just enabled

47.7.115 Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Write: POC:config or MB_DIS

This register contains message buffer configuration data for the transmission mode, the channel assignment, and for the cycle counter filtering. For detailed information on cycle counter filtering, refer to Message Buffer Cycle Counter Filtering .

NOTE

If at least one message buffer assigned to a certain slot is assigned to both channels, then all message buffers assigned to this slot have to be assigned to both channels. Otherwise, the

message buffer configuration is illegal and the result of the message buffer search is not defined.

Table 47-17. Channel Assignment Description

CHA	CHB	Transmit Message Buffer		Receive Message Buffer	
		static segment	dynamic segment	static segment	dynamic segment
1	1	transmit on both channel A and channel B	transmit on channel A only	store first valid frame received on either channel A or channel B	store first valid frame received on channel A, ignore channel B
0	1	transmit on channel B	transmit on channel B	store first valid frame received on channel B	store first valid frame received on channel B
1	0	transmit on channel A	transmit on channel A	store first valid frame received on channel A	store first valid frame received on channel A
0	0	no frame transmission	no frame transmission	no frame stored	no frame stored

Address: 0h base + 802h offset + (8d × i), where i=0d to 127d

Bit	0	1	2	3	4	5	6	7
Read Write	MTM	CHA	CHB	CCFE	CCFMSK			
Reset	0*	0*	0*	0*	0*	0*	0*	0*

Bit	8	9	10	11	12	13	14	15
Read Write	CCFMSK		CCFVAL					
Reset	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBCCFR*n* field descriptions

Field	Description
0 MTM	Message Buffer Transmission Mode. This control bit applies only to transmit message buffers and defines the transmission mode. 0 Event transmission mode 1 State transmission mode
1 CHA	Channel Assignment. In conjunction with the CHB field, defines the channel assignment and control the receive and transmit behavior of the message buffer according to Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*) .
2 CHB	Channel Assignment. In conjunction with the CHB field, defines the channel assignment and control the receive and transmit behavior of the message buffer according to Message Buffer Cycle Counter Filter Register (FR_MBCCFR*n*) .
3 CCFE	Cycle Counter Filtering Enable. This control bit is used to enable and disable the cycle counter filtering. 0 Cycle counter filtering disabled 1 Cycle counter filtering enabled

Table continues on the next page...

FR_MBCCFR*n* field descriptions (continued)

Field	Description
4–9 CCFMSK	Cycle Counter Filtering Mask. This field defines the filter mask for the cycle counter filtering.
10–15 CCFVAL	Cycle Counter Filtering Value. This field defines the filter value for the cycle counter filtering.

47.7.116 Message Buffer Frame ID Register (FR_MBFIDR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Write: POC:config or MB_DIS

Address: 0h base + 804h offset + (8d × i), where i=0d to 127d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0					FID										
Write																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBFIDR*n* field descriptions

Field	Description
0–4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–15 FID	Frame ID. The semantic of this field depends on the message buffer transfer type. • Receive Message Buffer: This field is used as a filter value to determine if the message buffer is used for reception of a message received in a slot with the slot ID equal to FID. • Transmit Message Buffer: This field is used to determine the slot in which the message in this message buffer should be transmitted.

47.7.117 Message Buffer Index Register (FR_MBIDXR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Write: POC:config or MB_DIS

Address: 0h base + 806h offset + (8d × i), where i=0d to 127d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	0								MBIDX							
Write																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBIDXR*n* field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–15 MBIDX	Message Buffer Index. This field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer. The application writes the index of the initially associated message buffer header field into this register. The CC updates this register after frame reception or transmission. Legal Values are 0 <= i <= 131 . Illegal values will be detected during the message buffer search.

47.7.118 Message Buffer Data Field Offset Register (FR_MBDOR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Write: POC:config

Address: 0h base + 1000h offset + (2d × i), where i=0d to 131d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	MBDO															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_MBDOR*n* field descriptions

Field	Description
0–15 MBDO	Message Buffer Data Field Offset. This field provides the data field offset belonging to a particular Message Buffer Index. For configuration constraints see Configure Data Field Offsets .

47.7.119 LRAM ECC Error Test Register (FR_LEETR*n*)

NOTE

After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

Address: 0h base + 1108h offset + (2d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	LEETD															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- After Phase3 reset, LRAM is not initialized. LRAM is initialized when CC leaves the Disabled Mode (For details, refer CHI LRAM Initialization section).

FR_LEETR*n* field descriptions

Field	Description
0–15 LEETD	LRAM ECC Error Test Data. This field contains the LRAM data belonging to the test register located in LRAM Bank n.

47.8 Functional Description

This section provides a detailed description of the functionality implemented in the CC.

47.8.1 Message Buffer Concept

The CC uses a data structure called a *message buffer* to store frame data, configuration, control, and status data. Each message buffer consists of two parts, the *message buffer control data* and the *physical message buffer*. The message buffer control data are located in dedicated registers. The structure of the message buffer control data depends on the message buffer type and is described in the Message Buffer Types section. The physical message buffer is located in the FlexRay memory area and is described in the next section.

47.8.2 Physical Message Buffer

All FlexRay messages and related frame and slot status information of received frames and of frames to be transmitted to the FlexRay bus are stored in data structures called *physical message buffers*. The physical message buffers are located in the FlexRay memory area.The structure of a physical message buffer is depicted in the figure below.

A physical message buffer consists of two fields, the *message buffer header field* and the *message buffer data field*. The message buffer header field contains the *frame header* and the *slot status*. The message buffer data field contains the *frame data*.

The connection between the two fields is established by the *data field offset*.

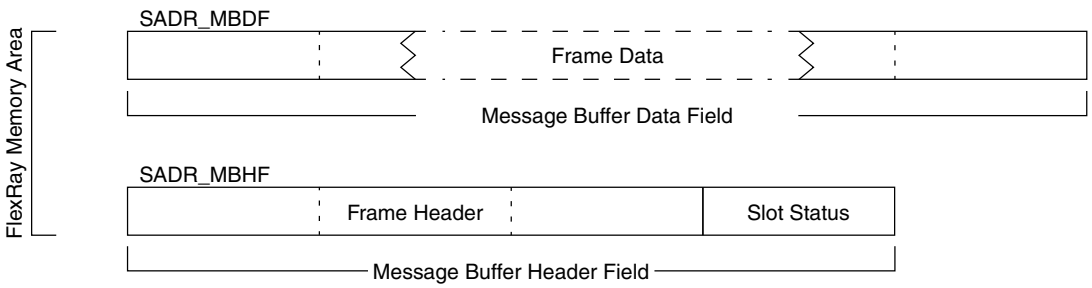


Figure 47-2. Physical Message Buffer Structure

47.8.2.1 Message Buffer Header Field

The message buffer header field is a contiguous region in the FlexRay memory area and occupies eight bytes. It contains the frame header, and the slot status. Its structure is shown in the above figure. The physical start address *SADR_MBHF* of the message buffer header field must be 16-bit aligned.

47.8.2.1.1 Frame Header

The frame header occupies the first six bytes in the message buffer header field. It contains all FlexRay frame header related information according to the *FlexRay Communications System Protocol Specification, Version*2.1 Rev A. A detailed description of the usage and the content of the frame header is provided in Frame Header Description section.

47.8.2.1.2 Slot Status

The slot status occupies the last two bytes of the message buffer header field. It provides the slot and frame status related information according to the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. A detailed description of the content and usage of the slot status is provided in the Slot Status Description section.

47.8.2.2 Message Buffer Data Field

The message buffer data field is a contiguous area of 2-byte entities. This field contains the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum length of this field depends on the specific message buffer configuration and is specified in the message buffer descriptions given in the following section.

47.8.3 Message Buffer Types

The CC provides three different types of message buffers.

- *Individual Message Buffers*
- *Receive Shadow Buffers*
- *Receive FIFO Buffers*

For each message buffer type the structure of the physical message buffer is identical. The message buffer types differ only in the structure and content of message buffer control data, which control the related physical message buffer. The message buffer control data are described in the following sections.

47.8.3.1 Individual Message Buffers

The individual message buffers are used for all types of frame transmission and for dedicated frame reception based on individual filter settings for each message buffer. The CC supports three types of individual message buffers, which are described in the Individual Message Buffer Functional Description section.

Each individual message buffer consists of two parts, the physical message buffer, which is located in the FlexRay memory area, and the message buffer control data, which are located in dedicated registers. The structure of an individual message buffer is given in the figure below.

Each individual message buffer has a message buffer number n assigned, which determines the set of message buffer control registers associated to this individual message buffer. The individual message buffer with message buffer number *n* is controlled by the registers FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, and FR_MBIDXRn.

The connection between the message buffer control registers and the physical message buffer is established by the message buffer index field MBIDX in the Message Buffer Index Registers (FR_MBIDXRn). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to this equation:

$$SADR_MBHF = (FR_MBIDXRn[MBIDX] \times 8) + SMBA$$

Equation 9

The data field belonging to a particular physical message buffer is characterized by the data field offset. For each physical message buffer with MBIDX i the FR_MBDORi contains the offset of the corresponding message buffer data field with respect to the CC FlexRay memory area base address as provided by SMBA field in the System Memory Base Address Register (FR_SYMBADR).

The data field offset is used to determine the start address SADR_MBDF of the corresponding message buffer data field in the FlexRay memory area according to this equation:

$$SADR_MBDF = [DataFieldOffset] + SMBA$$

Equation 10

The FR_MBDORn are stored in the module internal memory LRAM. Refer to CHI LRAM Initialization for the setup of the data field offset values.

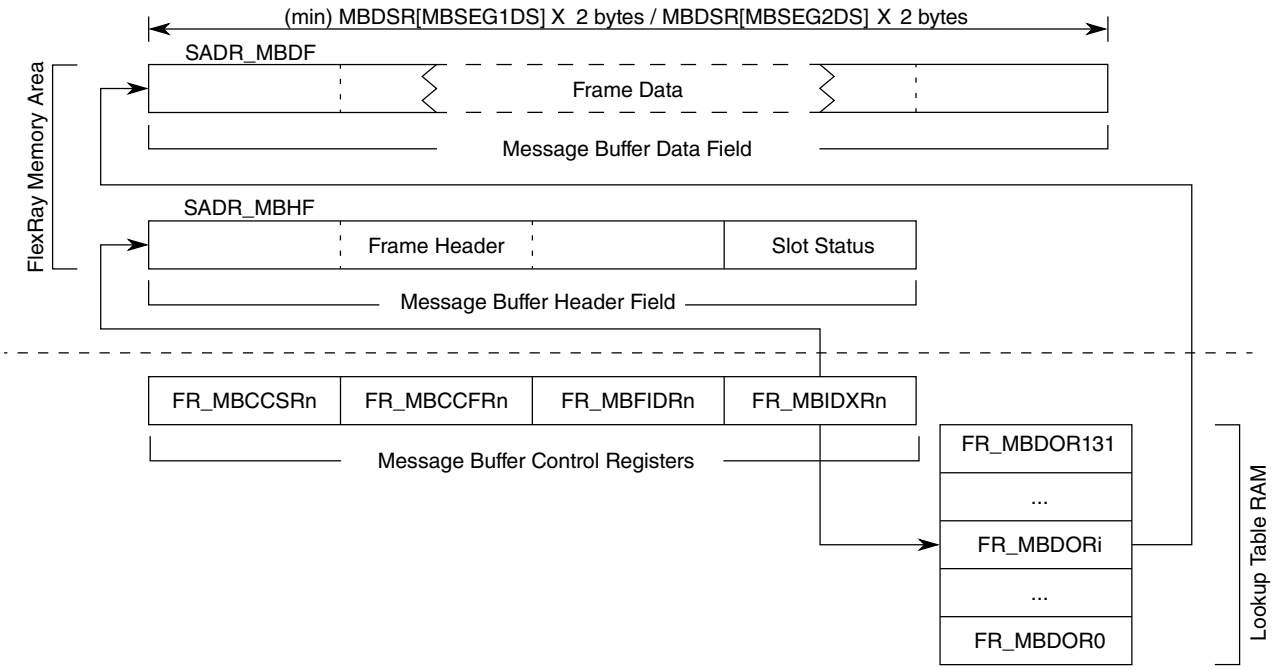


Figure 47-3. Individual Message Buffer Structure

47.8.3.1.1 Individual Message Buffer Segments

The set of the individual message buffers can be split up into two message buffer segments using the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR). All individual message buffers with a message buffer number n ≤ FR_MBSSUTR[LAST_MB_SEG1] belong to the first message buffer segment. All individual message buffers with a message buffer number n > FR_MBSSUTR[LAST_MB_SEG1] belong to the second message buffer segment. The following rules apply to the length of the message buffer data field:

- *all physical message buffers associated to individual message buffers that belong to the same message buffer segment must have message buffer data fields of the same length*
- *the minimum length of the message buffer data field for individual message buffers in the first message buffer segment is 2 × FR_MBDSR[MBSEG1DS] bytes*
- *the minimum length of the message buffer data field for individual message buffers assigned to the second segment is 2 × FR_MBDSR[MBSEG2DS] bytes.*

47.8.3.2 Receive Shadow Buffers

The receive shadow buffers are required for the frame reception process for individual message buffers. The CC provides four receive shadow buffers, one receive shadow buffer per channel and per message buffer segment.

Each receive shadow buffer consists of two parts, the physical message buffer located in the FlexRay memory area and the receive shadow buffer control registers located in dedicated registers. The structure of a receive shadow buffer is shown in the figure below. The four internal shadow buffer control registers can be accessed by the Receive Shadow Buffer Index Register (FR_RSBIR).

The connection between the receive shadow buffer control register and the physical message buffer for the selected receive shadow buffer is established by the receive shadow buffer index field RSBIDX in the Receive Shadow Buffer Index Register (FR_RSBIR). The start address SADR_MBHF of the related message buffer header field in the FlexRay memory area is determined according to this equation:

$$SADR_MBHF = (FR_RSBIR[RSBIDX] \times 8) + SMBA$$

Equation 11

The length required for the message buffer data field depends on the message buffer segment that the receive shadow buffer is assigned to. For the receive shadow buffers assigned to the first message buffer segment, the length must be the same as for the individual message buffers assigned to the first message buffer segment. For the receive shadow buffers assigned to the second message buffer segment, the length must be the same as for the individual message buffers assigned to the second message buffer segment. The receive shadow buffer assignment is described in Receive Shadow Buffer Index Register (FR_RSBIR).

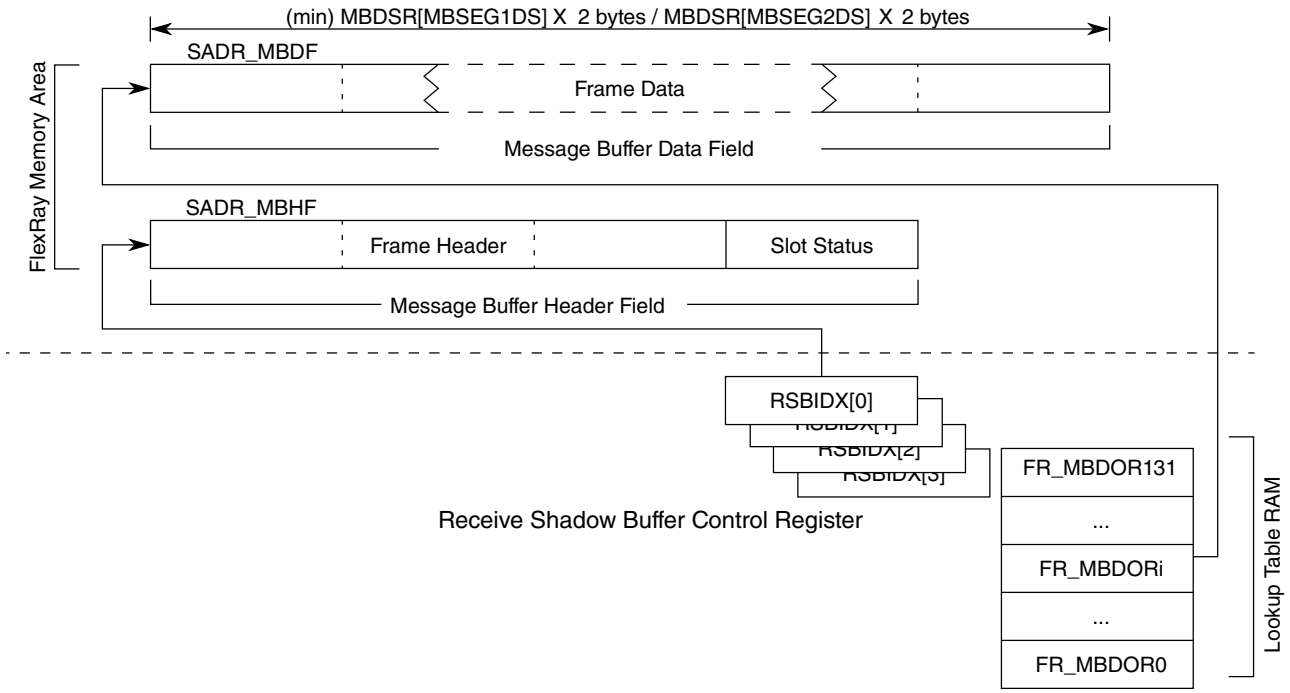


Figure 47-4. Receive Shadow Buffer Structure

47.8.3.3 Receive FIFO

The receive FIFO implements a frame reception system based on the FIFO concept. The CC provides two independent receive FIFOs, one per channel.

A receive FIFO consists of a set of physical message buffers in the FlexRay memory area and a set of receive FIFO control registers located in dedicated registers. The structure of a receive FIFO is given in the Receive FIFO Structure figure, later in this section.

The connection between the receive FIFO control registers and the set of physical message buffers is established by the Receive FIFO Start Index Register (FR_RFSIR), the Receive FIFO Depth and Size Register (FR_RFDSR), and the Receive FIFO A Read Index Register (FR_RFARIR) / Receive FIFO B Read Index Register (FR_RFBRIR).

The system memory base address SMBA valid for the receive FIFOs is defined by the system memory base address register selected by the FIFO address mode bit FR_MCR[FAM] in the Module Configuration Register.

The start byte address SADR_MBHF[1] of the first message buffer header field that belongs to the receive FIFO is determined according to Equation 12 on page 1984.

$$SADR_MBHF[1] = (8 \times FR_RFSIR[SIDX]) + SMBA$$

Equation 12

The start byte address SADR_MBHF[n] of the last message buffer header field that belongs to the receive FIFO in the FlexRay memory area is determined according Equation 13 on page 1985 .

$$SADR_MBHF[n] = (8 \times (FR_RFSIR[SIDX] + FR_RFDSR[FIFO_DEPTH])) + SMBA$$

Equation 13

The required information to access the current entry of the FIFO is given in the following registers:

- *The registers Receive FIFO A Read Index Register (FR_RFARIR) and Receive FIFO B Read Index Register (FR_RFBRIR) provide the index of the physical message buffer belonging to the current entry.*

  The data field offset belonging to the current FIFO entry RF_DFO[X] must be calculated using the current read index i according to the following formula:

$$RF_DFO[X] = FR_RFSDOR[X] + (FR_RFDSR[X][ENTRY_SIZE] \times 2) \times i - FR_RFSIDX[X])$$

Equation 14

Note

The current read index loops up starting at the number given in the FR_RD[A/B]RDIDX register for the required number of entries.

Refer to FIFO Update for details about updating the FIFO read pointer.

All message buffer header fields assigned to a receive FIFO are within a contiguous region defined by FR_RFSIR[SIDX] and FR_RFDSR[FIFO_DEPTH].

The data sections of all FIFO entries within on receive FIFO are of the same length defined by FR_RFDSR[FIFO_SIZE].

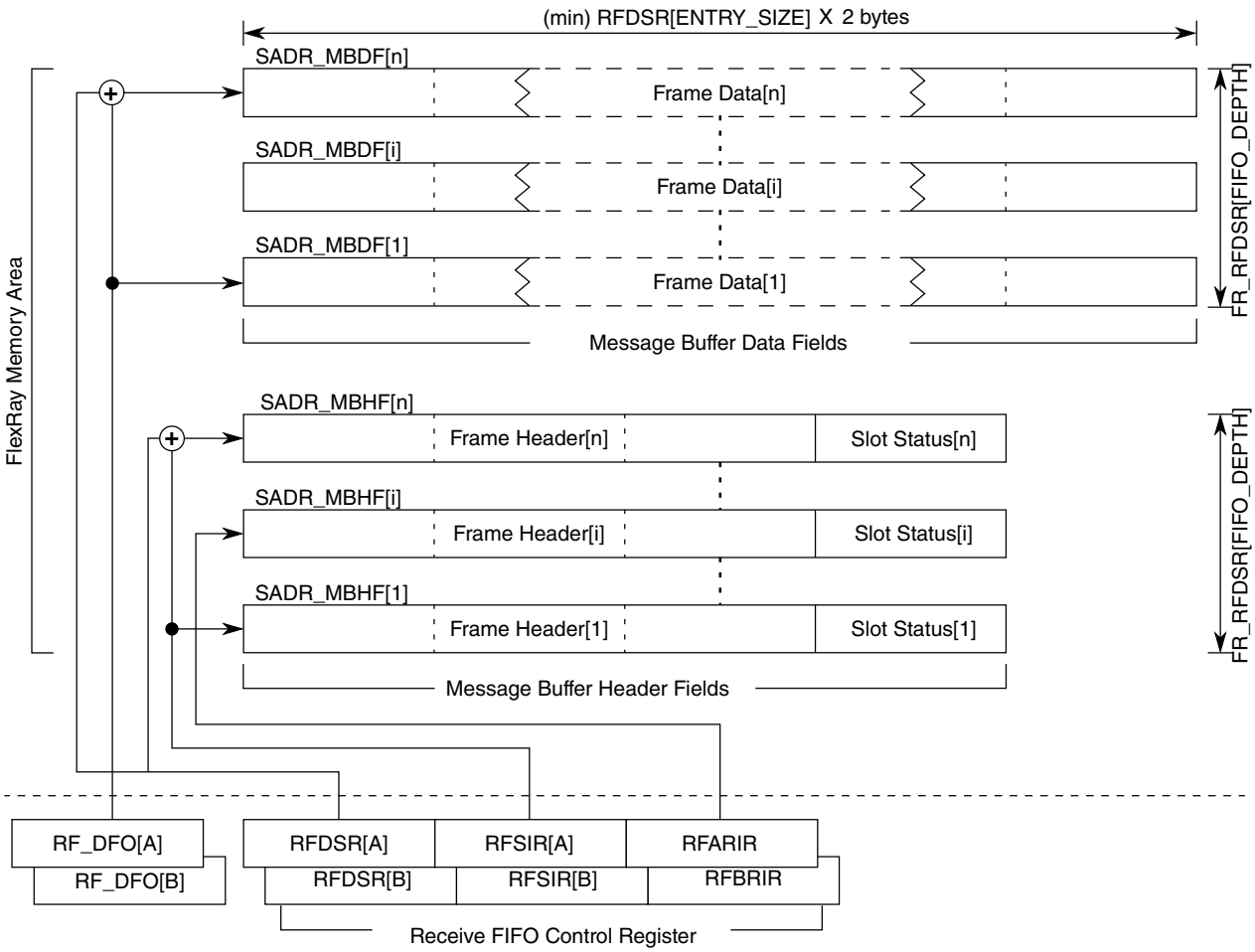


Figure 47-5. Receive FIFO Structure

Note

The actual values of the data field offsets RF_DFO[A/B] need to be calculated according to Equation 14 on page 1985. They are not stored in a register.

47.8.3.4 Message Buffer Configuration and Control Data

This section describes the configuration and control data for each message buffer type.

47.8.3.4.1 Individual Message Buffer Configuration Data

Before an individual message buffer can be used for transmission or reception, it must be configured. There is a set of common configuration parameters that applies to all individual message buffers and a set of configuration parameters that applies to each message buffer individually.

47.8.3.4.1.1 Common Configuration Data

The set of common configuration data for individual message buffers is located in the following registers.

- *Message Buffer Data Size Register (FR_MBDSR)*

  The MBSEG2DS and MBSEG1DS fields define the minimum length of the message buffer data field with respect to the message buffer segment.

- *Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)*

  The LAST_MB_SEG1 and LAST_MB_UTIL fields define the segmentation of the individual message buffers and the number of individual message buffers that are used. For more details, see Individual Message Buffer Segments.

47.8.3.4.1.2 Specific Configuration Data

The set of message buffer specific configuration data for individual message buffers is located in the following registers.

- *Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn)*

  The MTD bit configures the message buffer type.

- *Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn)*

  The MTM, CHA, CHB bits configure the transmission mode and the channel assignment. The CCFE, CCFMSK, and CCFVAL bits and fields configure the cycle counter filter.

- *Message Buffer Frame ID Registers (FR_MBFIDRn)*

  For a transmit message buffer, the FID field is used to determine the slot in which the message in this message buffer will be transmitted.

- *Message Buffer Index Registers (FR_MBIDXRn)This MBIDX field provides the index of the message buffer header field of the physical message buffer that is currently associated with this message buffer.*

47.8.3.5 Individual Message Buffer Control Data

During normal operation, each individual message buffer can be controlled by the control and trigger bits CMT, LCKT, EDT, and MBIE in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn).

47.8.3.6 Receive Shadow Buffer Configuration Data

Before frame reception into the individual message buffers can be performed, the receive shadow buffers must be configured. The configuration data are provided by the Receive Shadow Buffer Index Register (FR_RSBIR). For each receive shadow buffer, the application provides the message buffer header index. When the protocol is in the *POC:normal active* or *POC:normal passive* state, the receive shadow buffers are under full CC control.

47.8.3.7 Receive FIFO Control and Configuration Data

This section describes the configuration and control data for the two receive FIFOs.

47.8.3.7.1 Receive FIFO Configuration Data

The CC provides two functional independent receive FIFOs, one per channel. The FIFOs have a common subset of configuration data:

- *Receive FIFO Periodic Timer Register (FR_RFPTR)*

Each FIFO has its own set of configuration data. The configuration data are located in the following registers:

- *Receive FIFO Watermark and Selection Register (FR_RFWMSR)*
- *Receive FIFO Start Index Register (FR_RFSIR)*
- *Receive FIFO Start Data Offset Register (FR_RFSDOR)*
- *Receive FIFO Depth and Size Register (FR_RFDSR)*
- *Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR)*
- *Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR)*
- *Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR)*

- *Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR)*
- *Receive FIFO Range Filter Configuration Register (FR_RFRFCFR)*

47.8.3.7.2 Receive FIFO Control Data

The application can access the FIFOs at any time using the control bits in the following registers:

- *Global Interrupt Flag and Enable Register (FR_GIFER)*
- *Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)*

47.8.3.7.3 Receive FIFO Status Data

The current status of the receive fifo is provided in the following register:

- *Global Interrupt Flag and Enable Register (FR_GIFER)*
- *Receive FIFO A Read Index Register (FR_RFARIR)*
- *Receive FIFO B Read Index Register (FR_RFBRIR)*
- *Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR)*

47.8.4 FlexRay Memory Area Layout

The CC supports a wide range of possible layouts for the FlexRay memory area. Two basic layout modes can be selected by the FIFO address mode bit FR_MCR[FAM].

47.8.4.1 FlexRay Memory Area Layout (FR_MCR[FAM] = 0)

In the figure given below, shows an example layout for the FIFO address mode FR_MCR[FAM]=0. In this mode, the following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area is one contiguous region.
- The FlexRay memory area size is maximum 64 KB.
- The FlexRay memory area starts at a 16 byte boundary

The FlexRay memory area contains three areas: the *message buffer header area*, the *message buffer data area*, and the *sync frame table area*.

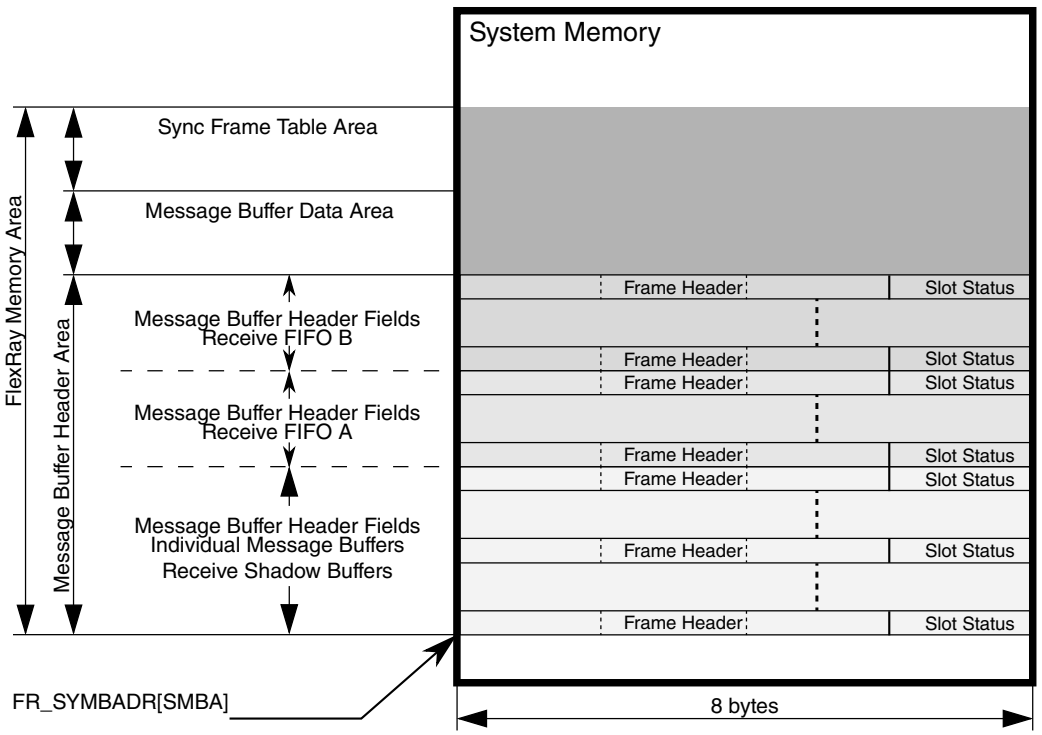


Figure 47-6. Example of FlexRay Memory Area Layout (FR_MCR[FAM] = 0)

47.8.4.2 FlexRay Memory Area Layout (FR_MCR[FAM] = 1)

The following figure shows an example layout for the FIFO address mode FR_MCR[FAM]=1. The following set of rules applies to the layout of the FlexRay memory area:

- The FlexRay memory area consists of two contiguous regions.
- The size of each region is maximum 64 KB.
- Each region start at a 16 byte boundary.

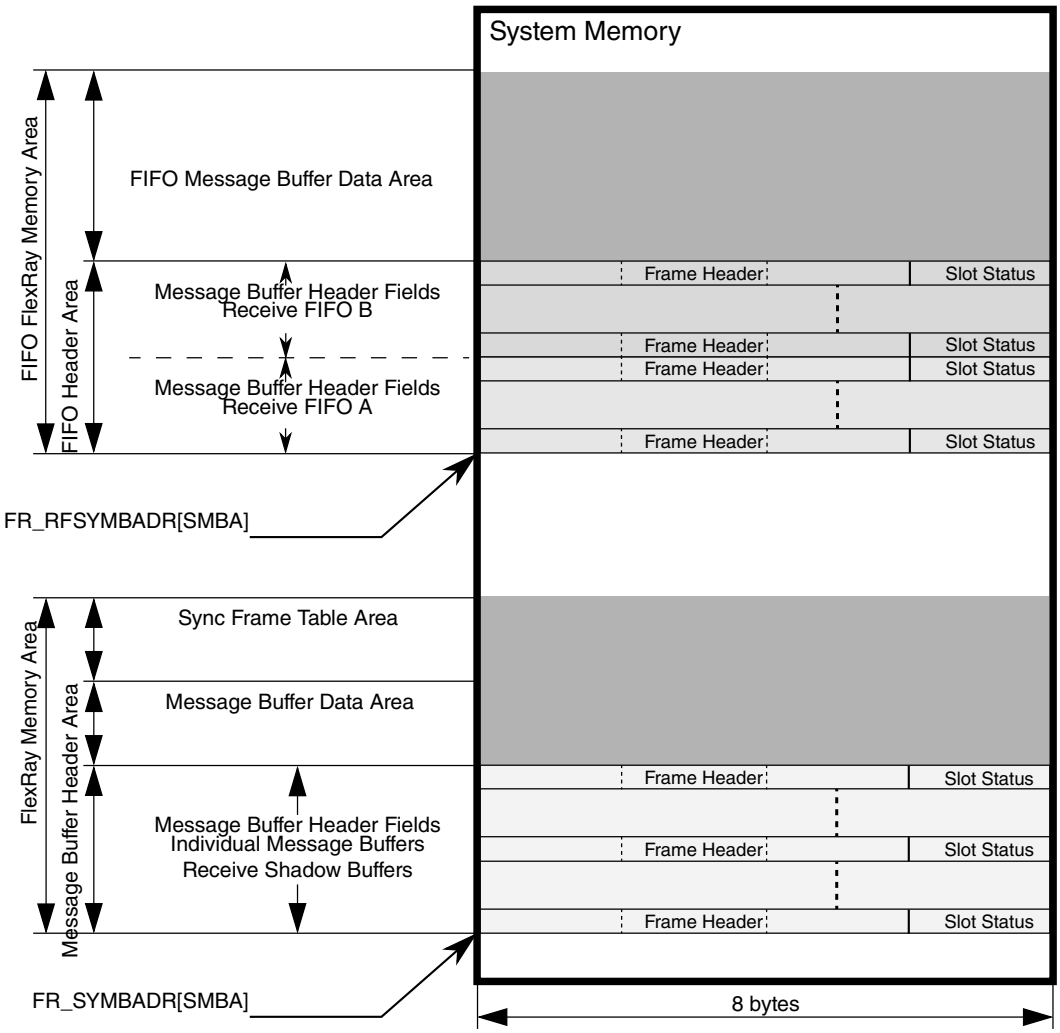


Figure 47-7. Example of FlexRay Memory Area Layout (FR_MCR[FAM] = 1)

47.8.4.3 Message Buffer Header Area (FR_MCR[FAM] = 0)

The message buffer header area contains all message buffer header fields of the physical message buffers for all message buffer types. The following rules apply to the message buffer header fields for the three type of message buffers.

1. The start byte address SADR_MBHF of each message buffer header field for *individual message buffers* and *receive shadow buffers* must fulfill Equation 15 on page 1991.

$$SADR_MBHF = (i \times 8) + FR_SYMBADR[SMBA]; (0 \le i \le 131)$$

Equation 15

2. The start byte address SADR_MBHF of each message buffer header field for the *FIFO* must fulfill Equation 15 on page 1991.

$$SADR_MBHF = (i \times 8) + FR_SYMBADR[SMBA]; (0 \le i \le 1023)$$

Equation 16

$$SADR_MBHF = (i \times 8) + FR_SYMBADR[SMBA]; (0 \le i \le 1023)$$

Equation 17

3. The message buffer header fields for each FIFO have to be a contiguous area.

47.8.4.4 Message Buffer Header Area (FR_MCR[FAM] = 1)

The message buffer header area contains all message buffer header fields of the physical message buffers for the individual message buffers and receiver shadow buffers. The following rules apply to the message buffer header fields for the two type of message buffers.

1. The start address SADR_MBHF of each message buffer header field for individual message buffers and receive shadow buffers must fulfill Equation 18 on page 1992.

$$SADR_MBHF = (i \times 8) + FR_SYMBADR[SMBA]; (0 \le i \le 131)$$

Equation 18

47.8.4.5 FIFO Message Buffer Header Area (FR_MCR[FAM] = 1)

The FIFO message buffer header area contains all message buffer header fields of the physical message buffers for the FIFO. The following rules apply to the FIFO message buffer header fields.

1. The start byte address SADR_MBHF of each message buffer header field for the FIFO must fulfill the following equation:Equation 19 on page 1992.

$$SADR_MBHF = (i\gamma 8) + FR_RFSYMBADR[SMBA]; (0 \le i \le 1023)$$

Equation 19

2. The message buffer header fields for each FIFO have to be a contiguous area.

47.8.4.6 Message Buffer Data Area

The message buffer data area contains all the message buffer data fields of the physical message buffers. Each message buffer data field must start at a 16-bit boundary.

47.8.4.7 Sync Frame Table Area

The sync frame table area is used to provide a copy of the internal sync frame tables for application access. Refer to Sync Frame ID and Sync Frame Deviation Tables for the description of the sync frame table area.

47.8.5 Physical Message Buffer Description

This section provides a detailed description of the usage and the content of the two parts of a physical message buffer, the message buffer header field and the message buffer data field.

47.8.5.1 Message Buffer Protection and Data Consistency

The physical message buffers are located in the FlexRay memory area. The CC provides no means to protect the FlexRay memory area from uncontrolled or illegal host or other client write access. To ensure data consistency of the physical message buffers, the application must follow the write access scheme that is given in the description of each of the physical message buffer fields.

47.8.5.2 Message Buffer Header Field Description

This section provides a detailed description of the usage and content of the message buffer header field. A description of the structure of the message buffer header fields is given in Message Buffer Header Field. Each message buffer header field consists of two sections: the frame header section and the slot status section.

47.8.5.2.1 Frame Header Description

Frame header content, access, and checks are discussed in this section.

47.8.5.2.1.1 Frame Header Content

The semantic and content of the frame header section depends on the message buffer type.

For individual receive message buffers and receive FIFOs, the frame header receives the frame header data of the *first valid frame* received on the assigned channels.

For receive shadow buffers, the frame header receives the frame header data of the current frame received regardless of whether the frame is valid or not.

For transmit message buffers, the application writes the frame header of the frame to be transmitted into this location. The frame header will be read out when the frame is transferred to the FlexRay bus.

The structure of the frame header in the message buffer header field for receive message buffers and the receive FIFO is given in the figure below. A detailed description is given in Table 47-22.

Table 47-18. Frame Header Structure (Receive Message Buffer and Receive FIFO)

0x0	R	PPI	NUF	SYF	SUF	FID		
0x2	0	0	CYCCNT			0	PLDLEN	
0x4	0	0	0	0	0	HDCRC		

The structure of the frame header in the message buffer header field for transmit message buffers is given in the following figure. A detailed description is given in Table 47-23. The checks that will be performed are described in Frame Header Checks.

Table 47-19. Frame Header Structure (Transmit Message Buffer)

0x0	R	PPI	NUF	SYF	SUF	FID [1]		
0x2			CYCCNT				PLDLEN [2]	
0x4						HDCRC		

= not used

1. checked
2. checked if not static

The structure of the frame header in the message buffer header field for transmit message buffers assigned to key slot is given in the figure below.

Table 47-20. Frame Header Structure (Transmit Message Buffer for Key Slot)

0x0	R	PPI	NUF	SYF	SUF	FID		
0x2			CYCCNT				PLDLEN	
0x4						HDCRC		

= not used

47.8.5.2.1.2 Frame Header Access

The frame header is located in the FlexRay memory area. To ensure data consistency, the application must follow the write access scheme described below.

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the frame header field.

For transmit message buffers, the application must follow the write access restrictions given in the following table. This table shows the condition under which the application can write to the frame header entries without corrupting the FlexRay message transmission.

Table 47-21. Frame Header Write Access Constraints (Transmit Message Buffer)

Field	Static Segment	Dynamic Segment
FID	*POC:config* or MB_DIS	
PPI, PLDLEN, HDCRC	*POC:config* or MB_DIS or	MB_LCK

47.8.5.2.1.3 Frame Header Checks

As shown in Table 47-19 and Table 47-20 not all fields in the message buffer frame header are used for transmission. Some fields in the message buffer frame header are ignored, some are used for transmission, and some of them are checked for correct values. All checks that will be performed are described below.

For message buffers assigned to the key slot, no checks will be performed.

The value of the FID field must be equal to the value of the corresponding Message Buffer Frame ID Registers (FR_MBFIDRn). If the CC detects a mismatch while transmitting the frame header, it will set the frame ID error flag FID_EF in the CHI Error Flag Register (FR_CHIERFR). The value of the FID field will be ignored and replaced by the value provided in the Message Buffer Frame ID Registers (FR_MBFIDRn).

For transmit message buffers assigned to the *static* segment, the PLDLEN value must be equal to the value of the payload_length_static field in the Protocol Configuration Register 19 (FR_PCR19). If this is not fulfilled, the static payload length error flag SPL_EF in the CHI Error Flag Register (FR_CHIERFR) is set when the message buffer is under transmission. A syntactically and semantically correct frame is generated with payload_length_static payload words and the payload length field in the transmitted frame header set to payload_length_static.

For transmit message buffers assigned to the *dynamic* segment, the PLDLEN value must be less than or equal to the value of the max_payload_length_dynamic field in the Protocol Configuration Register 24 (FR_PCR24). If this is not fulfilled, the dynamic payload length error flag DPL_EF in the CHI Error Flag Register (FR_CHIERFR) is set

when the message buffer is under transmission. A syntactically and semantically correct dynamic frame is generated with PLDLEN payload words and the payload length field in the frame header set to PLDLEN.

Table 47-22. Frame Header Field Descriptions (Receive Message Buffer and Receive FFO)

Field	Description
R	**Reserved Bit** — This is the value of the *Reserved bit* of the received frame stored in the message buffer
PPI	**Payload Preamble Indicator** — This is the value of the *Payload Preamble Indicator* of the received frame stored in the message buffer.
NUF	**Null Frame Indicator** — This is the value of the *Null Frame Indicator* of the received frame stored in the message buffer.
SYF	**Sync Frame Indicator** — This is the value of the *Sync Frame Indicator* of the received frame stored in the message buffer.
SUF	**Startup Frame Indicator** — This is the value of the *Startup Frame Indicator* of the received frame stored in the message buffer.
FID	**Frame ID** — This is the value of the *Frame ID* field of the received frame stored in the message buffer.
CYCCNT	**Cycle Count** — This is the number of the communication cycle in which the frame stored in the message buffer was received.
PLDLEN	**Payload Length** — This is the value of the *Payload Length* field of the received frame stored in the message buffer.
HDCRC	**Header CRC** — This is the value of the *Header CRC* field of the received frame stored in the message buffer.

Table 47-23. Frame Header Field Descriptions (Transmit Message Buffer)

Field	Description
R	**Reserved Bit** — This bit is not used, the value of the *Reserved bit* is generated internally according to *FlexRay Communications System Protocol Specification, Version*2.1 Rev A.
PPI	**Payload Preamble Indicator** — This bit provides the value of the *Payload Preamble Indicator* for the frame transmitted from the message buffer.
NUF	**Null Frame Indicator** — This bit is not used, the value of the *Null Frame Indicator* is generated internally according to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.
SYF	**Sync Frame Indicator** — This bit is not used, the value of the *Sync Frame Indicator* is generated internally according to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.
SUF	**Startup Frame Indicator** — This bit is not used, the value of the *Startup Frame Indicator* is generated internally according to *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.
FID	**Frame ID** — This field is checked as described in Frame Header Checks.
CYCCNT	**Cycle Count** — This field is not used, the value of the transmitted *Cycle Count* field is taken from the internal communication cycle counter.
PLDLEN	**Payload Length** — This field is checked and used as described in Frame Header Checks.
HDCRC	**Header CRC** — This field provides the value of the *Header CRC* field for the frame transmitted from the message buffer.

47.8.5.2.2 Slot Status Description

The slot status is a read-only structure for the application and a write-only structure for the CC. The meaning and content of the slot status in the message buffer header field depends on the message buffer type.

47.8.5.2.2.1 Receive Message Buffer and Receive FIFO Slot Status Description

This section describes the slot status structure for the individual receive message buffers and receive FIFOs. The content of the slot status structure for receive message buffers depends on the message buffer type and on the channel assignment for individual receive message buffers as given in the next table.

Table 47-24. Receive Message Buffer Slot Status Content

Receive Message Buffer Type	Slot Status Content
Individual Receive Message Buffer assigned to both channels FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=1	see Table 47-25
Individual Receive Message Buffer assigned to channel A FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=0	see Table 47-26
Individual Receive Message Buffer assigned to channel B FR_MBCCFRn[CHA]=0 and FR_MBCCFRn[CHB]=1	see Table 47-27
Receive FIFO Channel A Message Buffer	see Table 47-26
Receive FIFO Channel B Message Buffer	see Table 47-27

The meaning of the bits in the slot status structure is explained in the Receive Message Buffer Slot Status Field Description table below.

Table 47-25. Receive Message Buffer Slot Status Structure (ChAB)

R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	CH	VFA	SYA	NFA	SUA	SEA	CEA	BVA	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 47-26. Receive Message Buffer Slot Status Structure (ChA)

R	0	0	0	0	0	0	0	0	VFA	SYA	NFA	SUA	SEA	CEA	BVA	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 47-27. Receive Message Buffer Slot Status Structure (ChB)

R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	1	0	0	0	0	0	0	0	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 47-28. Receive Message Buffer Slot Status Field Description

Field	Description
Common Message Buffer Status Bits	
VFB	**Valid Frame on Channel B** — protocol related variable: *vSS!ValidFrame* channel B 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYB	**Sync Frame Indicator Channel B** — protocol related variable: *vRF!Header!SyFIndicator* channel B 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFB	**Null Frame Indicator Channel B** — protocol related variable: *vRF!Header!NFIndicator* channel B 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUB	**Startup Frame Indicator Channel B** — protocol related variable: *vRF!Header!SuFIndicator* channel B 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEB	**Syntax Error on Channel B** — protocol related variable: *vSS!SyntaxError* channel B 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEB	**Content Error on Channel B** — protocol related variable: *vSS!ContentError* channel B 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVB	**Boundary Violation on Channel B** — protocol related variable: *vSS!BViolation* channel B 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
CH	**Channel first valid received** — This status bit applies only to receive message buffers assigned to the static segment and to both channels. It indicates the channel that has received the *first valid* frame in the slot. This flag is set to 0 if no valid frame was received at all in the subscribed slot. 0 first valid frame received on channel A, or no valid frame received at all 1 first valid frame received on channel B
VFA	**Valid Frame on Channel A** — protocol related variable: *vSS!ValidFrame* channel A 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYA	**Sync Frame Indicator Channel A** — protocol related variable: *vRF!Header!SyFIndicator* channel A 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFA	**Null Frame Indicator Channel A** — protocol related variable: *vRF!Header!NFIndicator* channel A 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1

Table continues on the next page...

Table 47-28. Receive Message Buffer Slot Status Field Description (continued)

Field	Description
SUA	**Startup Frame Indicator Channel A** — protocol related variable: *vRF!Header!SuFIndicator* channel A 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEA	**Syntax Error on Channel A** — protocol related variable: *vSS!SyntaxError* channel A 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEA	**Content Error on Channel A** — protocol related variable: *vSS!ContentError* channel A 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVA	**Boundary Violation on Channel A** — protocol related variable: *vSS!BViolation* channel A 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1

47.8.5.2.2.2 Transmit Message Buffer Slot Status Description

This section describes the slot status structure for transmit message buffers. Only the TCA and TCB status bits are directly related to the transmission process. All other status bits in this structure are related to a receive process that may have occurred. The content of the slot status structure for transmit message buffers depends on the channel assignment as given in the table below.

Table 47-29. Transmit Message Buffer Slot Status Content

Transmit Message Buffer Type	Slot Status Content
Individual Transmit Message Buffer assigned to both channels FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=1	see Table 47-30
Individual Transmit Message Buffer assigned to channel A FR_MBCCFRn[CHA]=1 and FR_MBCCFRn[CHB]=0	see Table 47-31
Individual Transmit Message Buffer assigned to channel B FR_MBCCFRn[CHA]=0 and FR_MBCCFRn[CHB]=1	see Table 47-32

The meaning of the bits in the slot status structure is described in the Transmit Message Buffer Slot Status Structure Field Descriptions table.

Table 47-30. Transmit Message Buffer Slot Status Structure (ChAB)

R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 47-31. Transmit Message Buffer Slot Status Structure (ChA)

R	0	0	0	0	0	0	0	0	VFA	SYA	NFA	SUA	SEA	CEA	BVA	TCA
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 47-32. Transmit Message Buffer Slot Status Structure (ChB)

R	VFB	SYB	NFB	SUB	SEB	CEB	BVB	TCB	0	0	0	0	0	0	0	0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 47-33. Transmit Message Buffer Slot Status Structure Field Descriptions

Field	Description
VFB	**Valid Frame on Channel B** — protocol related variable: *vSS!ValidFrame* channel B 0 *vSS!ValidFrame* = 0 1 *vSS!ValidFrame* = 1
SYB	**Sync Frame Indicator Channel B** — protocol related variable: *vRF!Header!SyFIndicator* channel B 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFB	**Null Frame Indicator Channel B** — protocol related variable: *vRF!Header!NFIndicator* channel B 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUB	**Startup Frame Indicator Channel B** — protocol related variable: *vRF!Header!SuFIndicator* channel B 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEB	**Syntax Error on Channel B** — protocol related variable: *vSS!SyntaxError* channel B 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEB	**Content Error on Channel B** — protocol related variable: *vSS!ContentError* channel B 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVB	**Boundary Violation on Channel B** — protocol related variable: *vSS!BViolation* channel B 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
TCB	**Transmission Conflict on Channel B** — protocol related variable: *vSS!TxConflict* channel B 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1
VFA	**Valid Frame on Channel A** — protocol related variable: *vSS!ValidFrame* channel A 0 *vSS!ValidFrame* = 0

Table continues on the next page...

Table 47-33. Transmit Message Buffer Slot Status Structure Field Descriptions (continued)

Field	Description
	1 *vSS!ValidFrame* = 1
SYA	**Sync Frame Indicator Channel A** — protocol related variable: *vRF!Header!SyFIndicator* channel A 0 *vRF!Header!SyFIndicator* = 0 1 *vRF!Header!SyFIndicator* = 1
NFA	**Null Frame Indicator Channel A** — protocol related variable: *vRF!Header!NFIndicator* channel A 0 *vRF!Header!NFIndicator* = 0 1 *vRF!Header!NFIndicator* = 1
SUA	**Startup Frame Indicator Channel A** — protocol related variable: *vRF!Header!SuFIndicator* channel A 0 *vRF!Header!SuFIndicator* = 0 1 *vRF!Header!SuFIndicator* = 1
SEA	**Syntax Error on Channel A** — protocol related variable: *vSS!SyntaxError* channel A 0 *vSS!SyntaxError* = 0 1 *vSS!SyntaxError* = 1
CEA	**Content Error on Channel A** — protocol related variable: *vSS!ContentError* channel A 0 *vSS!ContentError* = 0 1 *vSS!ContentError* = 1
BVA	**Boundary Violation on Channel A** — protocol related variable: *vSS!BViolation* channel A 0 *vSS!BViolation* = 0 1 *vSS!BViolation* = 1
TCA	**Transmission Conflict on Channel A** — protocol related variable: *vSS!TxConflict* channel A 0 *vSS!TxConflict* = 0 1 *vSS!TxConflict* = 1

47.8.5.3 Message Buffer Data Field Description

The message buffer data field is used to store the frame payload data, or a part of it, of the frame to be transmitted to or received from the FlexRay bus. The minimum required length of this field depends on the message buffer type that the physical message buffer is assigned to and is given in the table below. The structure of the message buffer data field is given in the next figure.

Table 47-34. Message Buffer Data Field Minimum Length

physical message buffer assigned to	minimum length defined by
Individual Message Buffer in Segment 1	FR_MBDSR[MBSEG1DS]

Table continues on the next page...

Table 47-34. Message Buffer Data Field Minimum Length (continued)

physical message buffer assigned to	minimum length defined by
Receive Shadow Buffer in Segment 1	FR_MBDSR[MBSEG1DS]
Individual Message Buffer in Segment 2	FR_MBDSR[MBSEG2DS]
Receive Shadow Buffer in Segment 2	FR_MBDSR[MBSEG2DS]
Receive FIFO for channel A	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 0)
Receive FIFO for channel B	FR_RFDSR[ENTRY_SIZE] (FR_RFWMSR[SEL] = 1)

Note

The CC will not access any locations outside the message buffer data field boundaries given in the table above.

Table 47-35. Message Buffer Data Field Structure

0x0	DATA0 / MID0 / NMV0	DATA1 / MID1 / NMV1
0x2	DATA2 / NMV2	DATA3 / NMV3
…	…	…
0xN-2	DATA N-2	DATA N-1

The message buffer data field is located in the FlexRay memory area; thus, the CC has no means to control application write access to the field. To ensure data consistency, the application must follow a write and read access scheme.

47.8.5.3.1 Message Buffer Data Field Read Access

For transmit message buffers, the CC will not modify the content of the Message Buffer Data Field. Thus the application can read back the data at any time without any impact on data consistency.

For receive message buffers the application must lock the related receive message buffer and retrieve the message buffer header index from the Message Buffer Index Registers (FR_MBIDXRn). While the message buffer is locked, the CC will not update the Message Buffer Data Field.

For receive FIFOs, the application can read the message buffer indicated by the Receive FIFO A Read Index Register (FR_RFARIR) or the Receive FIFO B Read Index Register (FR_RFBRIR) when the related fill levels in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) indicate an non-empty FIFO.

47.8.5.3.2 Message Buffer Data Field Write Access

For receive message buffers, receive shadow buffers, and receive FIFOs, the application must not write to the message buffer data field.

For transmit message buffers, the application must follow the write access restrictions given in the Frame Data Write Access Constraints table below.

Table 47-36. Frame Data Write Access Constraints

Field	CC/MB State
DATA, MID, NMV	*POC:config* or MB_DIS or MB_LCK

Table 47-37. Frame Data Field Descriptions

Field	Description
DATA 0, DATA 1, … DATA N-1	**Message Data** — Provides the message data received or to be transmitted.For receive message buffer and receive FIFOs, this field provides the message data received for this message buffer. For transmit message buffers, the field provides the message data to be transmitted.
MID 0, MID 1	**Message Identifier** — If the payload preamble bit PPI is set in the message buffer frame header, the MID field holds the message ID of a dynamic frame located in the message buffer. The receive FIFO filter uses the received message ID for message ID filtering.
NMV 0, NMV 1, … NMV 11	**Network Management Vector** — If the payload preamble bit PPI is set in the message buffer frame header, the network management vector field holds the network management vector of a static frame located in the message buffer. **Note:** The MID and NMV bytes replace the corresponding DATA bytes.

47.8.6 Individual Message Buffer Functional Description

The CC provides these basic types of individual message buffers:

1. Transmit Message Buffers
2. Receive Message Buffers

Before an individual message buffer can be used, it must be configured by the application. After the initial configuration, the message buffer can be reconfigured later. The set of the configuration data for individual message buffers is given in Individual Message Buffer Configuration Data.

47.8.6.1 Individual Message Buffer Configuration

The individual message buffer configuration consists of two steps. The first step is the allocation of the required amount of memory for the FlexRay memory area. The second step is the programming of the message buffer configuration registers, which is described in this section.

47.8.6.1.1 Common Configuration Data

One part of the message buffer configuration data is common to all individual message buffers and the receive shadow buffers. These data can only be set when the protocol is in the *POC:config* state.

The application configures the number of utilized individual message buffers by writing the message buffer number of the last utilized message buffer into the LAST_MB_UTIL field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).

The application configures the size of the two segments of individual message buffers by writing the message buffer number of the last message buffer in the first segment into the LAST_MB_SEG1 field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).

The application configures the length of the message buffer data fields for both of the message buffer segments by writing to the MBSEG2DS and MBSEG1DS fields in the Message Buffer Data Size Register (FR_MBDSR).

Depending on the current receive functionality of the CC, the application must configure the receive shadow buffers. For each segment and for each channel with at least one individual receive message buffer assigned, the application must configure the related receive shadow buffer using the Receive Shadow Buffer Index Register (FR_RSBIR).

47.8.6.1.2 Specific Configuration Data

The second part of the message buffer configuration data is specific for each message buffer.

These data can be changed only when either

- *the protocol is in the POC:config state or*
- *the message buffer is disabled, i.e. FR_MBCCSRn[EDS] = 0*

The individual message buffer type is defined by the MTD and MBT bits in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn) as given in the following table.

Table 47-38. Individual Message Buffer Types

FR_MBCCSRn		Individual Message Buffer Description
MTD	MBT	
0	0	Receive Message Buffer
0	1	Reserved
1	0	Transmit Message Buffer
1	1	Reserved

The message buffer specific configuration data are

1. MTD bits in Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn)
2. all fields and bits in Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn)
3. all fields and bits in Message Buffer Frame ID Registers (FR_MBFIDRn)
4. all fields and bits in Message Buffer Index Registers (FR_MBIDXRn)

The meaning of the specific configuration data depends on the message buffer type, as given in the detailed message buffer type descriptions in the Transmit Message Buffers section and Receive Message Buffers section.

47.8.6.2 Transmit Message Buffers

The section provides a detailed description of the functionality of single buffered transmit message buffers.

A transmit message buffer is used by the application to provide message data to the CC that will be transmitted over the FlexRay Bus. The CC uses the transmit message buffers to provide information about the transmission process and status information about the slot in which message was transmitted.

The individual message buffer with message buffer number n is configured to be a transmit message buffer by the following settings:

- $FR__MBCCSRn[MBT] = 0(singlebufferedmessagebuffer)$
- $FR_MBCCSRn[MDT] = 1(transmitmessagebuffer)$

47.8.6.2.1 Access Regions

To certain message buffer fields, both the application and the CC have access. To ensure data consistency, a message buffer locking scheme is implemented, which is used to control the access to the data, control, and status bits of a message buffer. The access regions for transmit message buffers are depicted in the figure below. A description of the regions is given in Table 47-40. If an region is active as indicated in Table 47-41, the access scheme given for that region applies to the message buffer.

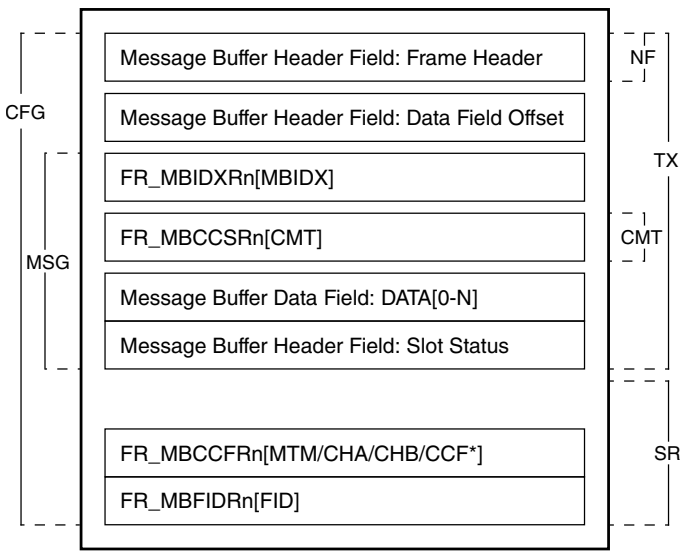


Figure 47-8. Transmit Message Buffer Access Regions

Table 47-39. Transmit Message Buffer Access Regions Description

Region	Access from		Region used for
	Application	Module	
CFG	read/write	-	Message Buffer Configuration
MSG	read/write	-	Message Data and Slot Status Access
NF	-	read-only	Message Header Access for Null Frame Transmission
TX	-	read/write	Message Transmission and Slot Status Update
CM	-	read-only	Message Buffer Validation
SR	-	read-only	Message Buffer Search

The trigger bits FR_MBCCSRn[EDT] and FR_MBCCSRn[LCKT], and the interrupt enable bit FR_MBCCSRn[MBIE] are not under access control and can be accessed from the application at any time. The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSRn[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC clear access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The transmit message buffer states are given in the next figure. A description of the states is given in Table 47-40, which also provides the access scheme for the access regions.

The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] provide the application with the required message buffer status information. The internal status information is not visible to the application.

47.8.6.2.2 Message Buffer States

This section describes the transmit message buffer states and provides a state diagram.

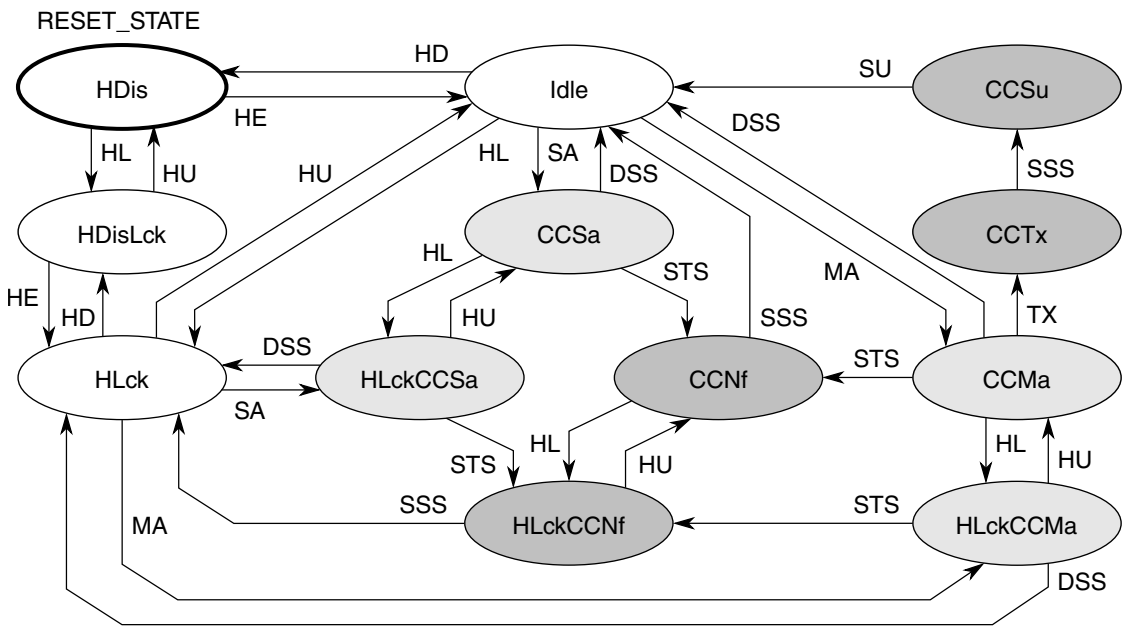


Figure 47-9. Transmit Message Buffer States

Table 47-40. Transmit Message Buffer State Description (Sheet 1 of 2)

State	FR_MBCCSRn		Access Region		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	–	CM,SR	**Idle** - Message Buffer is idle. Included in message buffer search.
HDis	0	0	CFG	–	**Disabled** - Message Buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	–	**Disabled and Locked** - Message Buffer under configuration.Excluded from message buffer search.
HLck	1	1	MSG	SR	**Locked** - Applications access to data, control, and status.Included in message buffer search.

Table continues on the next page...

Table 47-40. Transmit Message Buffer State Description (Sheet 1 of 2) (continued)

State	FR_MBCCSRn		Access Region		Description
	EDS	LCKS	Appl.	Module	
CCSa	1	0	–	–	**Slot Assigned** - Message buffer assigned to next static slot.Ready for Null Frame transmission.
HLckCCSa	1	1	MSG	–	**Locked and Slot Assigned** - Applications access to data, control, and status.Message buffer assigned to next static slot
CCNf	1	0	–	NF	**Null Frame Transmission** Header is used for null frame transmission.
HLckCCNf	1	1	MSG	NF	**Locked and Null Frame Transmission** - Applications access to data, control, and status. Header is used for null frame transmission.
CCMa	1	0	–	CM	**Message Available** - Message buffer is assigned to next slot and cycle counter filter matches.
HLckCCMa	1	1	MSG	–	**Locked and Message Available** - Applications access to data, control, and status. Message buffer is assigned to next slot and cycle counter filter matches.
CCTx	1	0	–	TX	**Message Transmission** - Message buffer data transmit. Payload data from buffer transmitted
CCSu	1	0	–	TX	**Status Update** - Message buffer status update. Update of status flags, the slot status field, and the header index.

47.8.6.2.3 Message Buffer Transitions

Application transitions, module transitions, and transition priorities are discussed in this section.

47.8.6.2.3.1 Application Transitions

The application transitions can be triggered by the application using the commands described in the table below. The application issues the commands by writing to the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

47.8.6.2.3.1.1 Message Buffer Enable and Disable

The enable and disable commands issued by writing 1 to the trigger bit FR_MBCCSRn[EDT]. The transition that will be triggered by each of these command depends on the current value of the status bit FR_MBCCSRn[EDS]. If the command triggers the disable transition HD and the message buffer is in one of the states CCSa,

HLckCCSa, CCMa, HLckCCMa, CCNf, HLckCCNf, or CCTx, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

47.8.6.2.3.1.2 Message Buffer Lock and Unlock

The lock and unlock commands issued by writing 1 to the trigger bit FR_MBCCSRn[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSRn[LCKS]. If the command triggers the lock transition HL and the message buffer is in the state CCTx, the lock transition has no effect (command is ignored) and message buffer state is not changed. In this case, the message buffer lock error flag LCK_EF in the CHI Error Flag Register (FR_CHIERFR) is set.

Table 47-41. Transmit Message Buffer Application Transitions

Transition	Command	Condition	Description
HE	FR_MBCCSRn[EDT]:= 1	FR_MBCCSRn[EDS] = 0	Application triggers message buffer enable.
HD		FR_MBCCSRn[EDS] = 1	Application triggers message buffer disable.
HL	FR_MBCCSRn[LCKT]:= 1	FR_MBCCSRn[LCKS] = 0	Application triggers message buffer lock.
HU		FR_MBCCSRn[LCKS] = 1	Application triggers message buffer unlock.

47.8.6.2.3.2 Module Transitions

The module transitions that can be triggered by the CC are described in the following table. Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 47-42. Transmit Message Buffer Module Transitions

Transition	Condition	Description
SA	slot match andstatic slot	**Slot Assigned** - Message buffer is assigned to next static slot.
MA	slot match andCycleCounter match	**Message Available** - Message buffer is assigned to next slot and cycle counter filter matches.
TX	slot start and FR_MBCCSRn[CMT] = 1	**Transmission Slot Start** - Slot Start and commit bit CMT is set. In case of a dynamic slot, pLatestTx is not exceeded.
SU	status updated	**Status Updated** - Slot Status field and message buffer status flags updated. Interrupt flag set.
STS	static slot start	**Static Slot Start** - Start of static slot.
DSS	dynamic slot start orsymbol window start orNIT start	**Dynamic Slot or Segment Start.** - Start of dynamic slot or symbol window or NIT.
SSS	slot start orsymbol window start orNIT start	**Slot or Segment Start** - Start of static slot or dynamic slot or symbol window or NIT.

47.8.6.2.3.3 Transition Priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in the first part of the next table, the module transitions have a higher priority than the application transitions. For all states except the CCMa state, both a lock/unlock transition HL/HD and a module transition can be executed at the same time. The result state is reached by first applying the application transition and subsequently the module transition to the intermediately reached state. For example, if the message buffer is in the HLck state and the application unlocks the message buffer by the HU transition and the module triggers the slot assigned transition SA, the intermediate state is Idle and the resulting state is CCSa.

The priorities among the module transitions is given in the second part of the table below.

Table 47-43. Transmit Message Buffer Transition Priorities

State	Priorities	Description
module vs. application		
Idle, HLck	SA > HD	Slot Assigned > Message Buffer Disable
	MA > HD	Message Available > Message Buffer Disable
CCMa	TX > HL	Transmission Start > Message Buffer Lock
module internal		
Idle, HLck	MA > SA	Message Available > Slot Assigned
CCMa	TX > STS	Transmission Slot Start > Static Slot Start
	TX > DSS	Transmission Slot Start > Dynamic Slot Start

47.8.6.2.4 Transmit Message Setup

To transmit a message over the FlexRay bus, the application writes the message data into the message buffer data field and sets the commit bit CMT in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). The physical access to the message buffer data field is described in Individual Message Buffers.

As indicated by Table 47-33, the application shall write to the message buffer data field and change the commit bit CMT only if the transmit message buffer is in one of the states HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa. The application can change the state of a message buffer if it issues the appropriate commands shown in Table 47-33. The state change is indicated through the FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] status bits.

If the transmit message buffer enters one of the states HDis, HDisLck, HLck, HLckCCSa, HLckCCMa, or HLckCCMa the FR_MBCCSRn[DVAL] flag is negated.

47.8.6.2.5 Message Transmission

As a result of the message buffer search described in Individual Message Buffer Search, the CC triggers the message available transition MA for up to two transmit message buffers. This changes the message buffer state from Idle to CCMa and the message buffers can be used for message transmission in the next slot.

The CC transmits a message from a message buffer if both of the following two conditions are fulfilled at the start of the transmission slot:

1. the message buffer is in the message available state CCMa
2. the message data are still valid, i.e. FR_MBCCSRn[CMT] = 1

In this case, the CC triggers the TX transition and changes the message buffer state to CCTx. A transmit message buffer timing and state change diagram for message transmission is given in the "Message Transmission Timing" figure below. In this example, the message buffer with message buffer number n is Idle at the start of the search slot, matches the slot and cycle number of the next slot, and message buffer data are valid, i.e. FR_MBCCSRn[CMT] = 1.

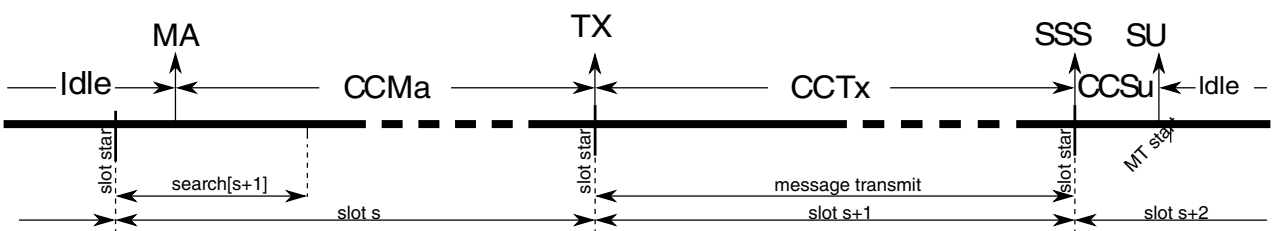


Figure 47-10. Message Transmission Timing

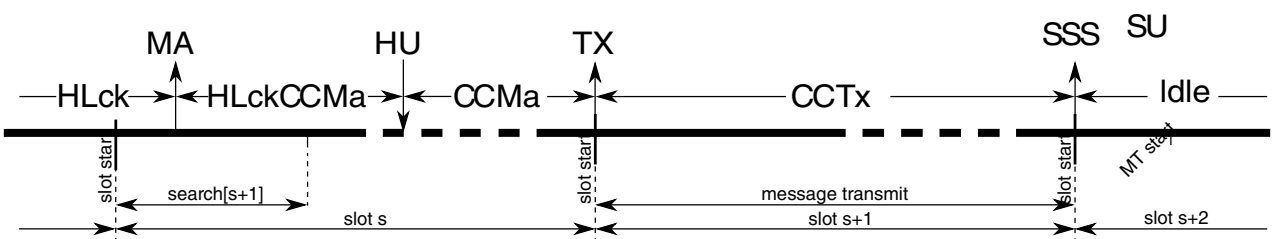


Figure 47-11. Message Transmission from HLck state with unlock

The amount of message data read from the FlexRay memory area and transferred to the FlexRay bus is determined by the following three items

1. the message buffer segment that the message buffer is assigned to, as defined by the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
2. the message buffer data field size, as defined by the related field of the Message Buffer Data Size Register (FR_MBDSR)
3. the value of the PLDLEN field in the message buffer header field, as described in Frame Header Description

If a message buffer is assigned to message buffer segment 1, and PLDLEN > MBSEG1DS, then 2 × MBSEG1DS bytes will be read from the message buffer data field and zero padding is used for the remaining bytes for the FlexRay bus transfer. If PLDLEN <= MBSEG1DS, the CC reads and transfers 2 ×PLDLEN bytes. The same holds for segment 2 and MBSEG2DS.

47.8.6.2.6 Null Frame Transmission

A static slot with slot number S is assigned to the CC for channel A, if at least one transmit message buffer is configured with the FR_MBFIDRn[FID] set to S and FR_MBCCFRn[CHA] set to 1. A Null Frame is transmitted in the static slot S on channel A, if this slot is assigned to the CC for channel A, and all transmit message buffers with FR_MBFIDRn[FID] = s and FR_MBCCFRn[CHA] = 1 are either not committed, i.e FR_MBCCSRn[CMT] = 0, or locked by the application, i.e. FR_MBCCSRn[LCKS] = 1, or the cycle counter filter is enabled and does not match.

Additionally, the application can clear the commit bit of a message buffer that is in the CCMa state, which is called *uncommit* or *transmit abort*. This message buffer will be used for null frame transmission.

As a result of the message buffer search described in Individual Message Buffer Search, the CC triggers the slot assigned transition SA for up to two transmit message buffers if at least one of the conditions mentioned above is fulfilled for these message buffers. The transition SA changes the message buffer states from either Idle to CCSa or from HLck to HLckCCSa. In each case, these message buffers will be used for null frame transmission in the next slot. A message buffer timing and state change diagram for null frame transmission from Idle state is given in the figure below.

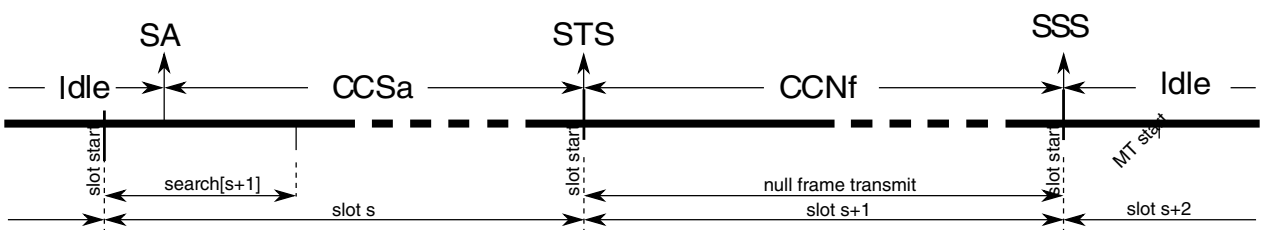


Figure 47-12. Null Frame Transmission from Idle state

A message buffer timing and state change diagram for null frame transmission from HLck state is given in the following figure.

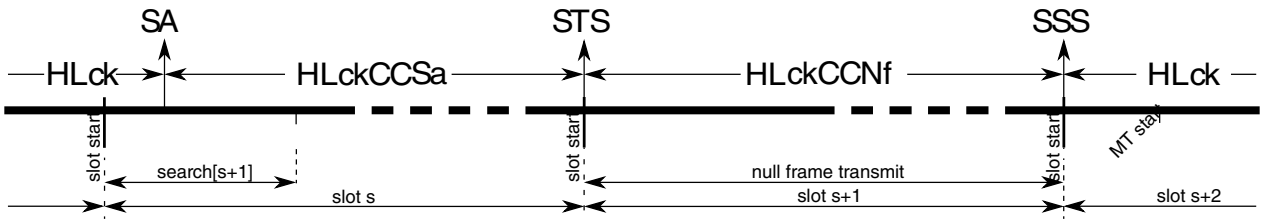


Figure 47-13. Null Frame Transmission from HLck state

If a transmit message buffer is in the CCSa or HLckCCSa state at the start of the transmission slot, a null frame is transmitted in any case, even if the message buffer is unlocked or committed before the transmission slot starts. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in the next figure.

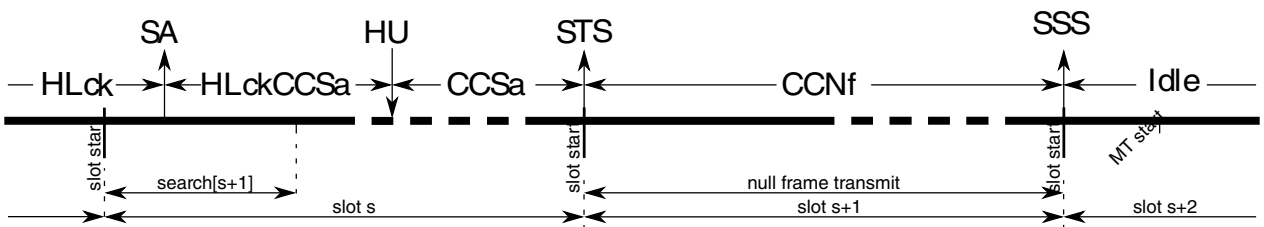


Figure 47-14. Null Frame Transmission from HLck state with unlock

Since the null frame transmission will not use the message buffer data, the application can lock/unlock the message buffer during null frame transmission. A transmit message buffer timing and state change diagram for null frame transmission for this case is given in the figure below.

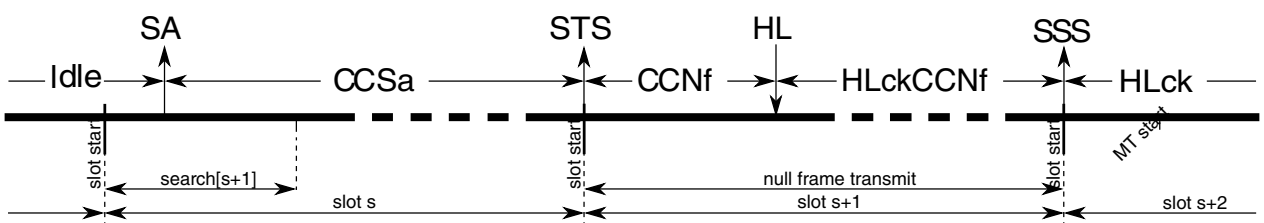


Figure 47-15. Null Frame Transmission from Idle state with locking

47.8.6.2.7 Message Buffer Status Update

After the end of each slot, the PE generates the slot status vector. Depending on the this status, the transmitted frame type, and the amount of transmitted data, the message buffer status is updated.

47.8.6.2.7.1 Message Buffer Status Update after Complete Message Transmission

The term complete message transmission refers to the fact that all payload data stored in the message buffer were send to FlexRay bus. In this case, the CC updates the slot status field of the message buffer and triggers the status updated transition SU. With the SU transition, the CC sets the message buffer interrupt flag FR_MBCCSRn[MBIF] to indicate the successful message transmission.

Depending on the transmission mode flag FR_MBCCFRn[MTM], the CC changes the commit flag FR_MBCCSRn[CMT] and the valid flag FR_MBCCSRn[DVAL]. If the FR_MBCCFRn[MTM] flag is negated, the message buffer is in the *event transmission mode*. In this case, each committed message is transmitted only once. The commit flag FR_MBCCSRn[CMT] is cleared with the SU transition. If the FR_MBCCFRn[MTM] flag is asserted, the message buffer is in the *state transmission mode*. In this case, each committed message is transmitted as long as the application provides new data or locks the message buffers. The CC will not clear the FR_MBCCSRn[CMT] flag at the end of transmission and will set the valid flag FR_MBCCSRn[DVAL] to indicate that the message will be transmitted again.

47.8.6.2.7.2 Message Buffer Status Update after Incomplete Message Transmission

The term incomplete message transmission refers to the fact that not all payload data that should be transmitted were send to FlexRay bus. This may be caused by the following regular conditions in the dynamic segment:

1. The transmission slot starts in a minislot with a minislot number greater than *pLatestTx*.
2. The transmission slot did not exist in the dynamic segment at all.

In any of these two aforementioned conditions occur, the status of the message buffer is not changed at all with the SU transition. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

Additionally, an incomplete message transmission can be caused by internal communication errors. If those errors occur, the Protocol Engine Communication Failure Interrupt Flag PECF_IF is set in the Protocol Interrupt Flag Register 1 (FR_PIFR1).

47.8.6.2.7.3 Message Buffer Status Update after Null Frame Transmission

After the transmission of a null frame, the status of the message buffer that was used for the null frame transmission is not changed at all. The slot status field is not updated, the status and control flags are not changed, and the interrupt flag is not set.

47.8.6.3 Receive Message Buffers

The section provides a detailed description of the functionality of the receive message buffers. If receive message buffers are used it is required to configure the related receive shadow buffer as described in Receive Shadow Buffers.

A receive message buffer is used to receive a message from the FlexRay Bus based on individual filter criteria. The CC uses the receive message buffer to provide the following data to the application.

1. message data received
2. information about the reception process
3. status information about the slot in which the message was received

A individual message buffer with message buffer number *n* is configured as a receive message buffer by the following configuration settings

$$FR_MBCCSRn[MTD] = 0 (receive message buffer)$$

Equation 20

To certain message buffer fields, both the application and the CC have access. To ensure data consistency, a message buffer locking scheme is implemented that is used to control the access to the data, control, and status bits of a message buffer. The access regions for receive message buffers are depicted in the figure below. A description of the regions is given in the following table. If a region is active as indicated in the "Receive Message Buffer States and Access (Sheet 2 of 2)" table below, the access scheme given for that region applies to the message buffer.

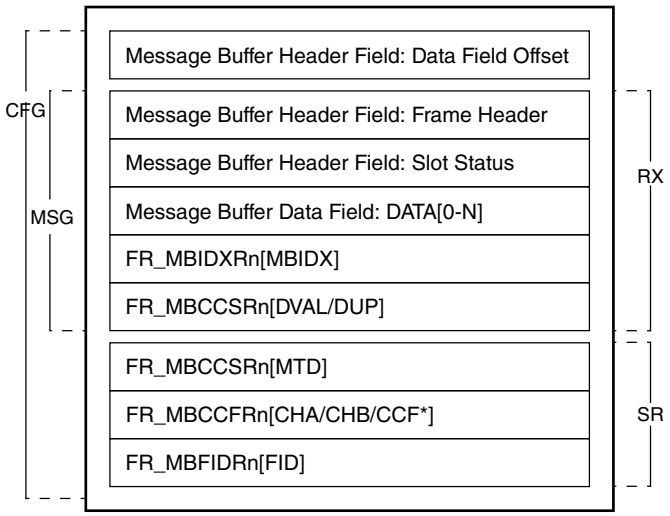


Figure 47-16. Receive Message Buffer Access Regions

Table 47-44. Receive Message Buffer Access Region Description

Region	Access from		Region used for
	Application	Module	
CFG	read/write	-	Message Buffer Configuration, Message Data and Status Access
MSG	read/write	-	Message Data, Header, and Status Access
RX	-	write-only	Message Reception and Status Update
SR	-	read-only	Message Buffer Search Data

The trigger bits FR_MBCCSRn[EDT] and FR_MBCCSRn[LCKT] and the interrupt enable bit FR_MBCCSRn[MBIE] are not under access control and can be accessed from the application at any time. The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] are not under access control and can be accessed from the CC at any time.

The interrupt flag FR_MBCCSRn[MBIF] is not under access control and can be accessed from the application and the CC at any time. CC set access has higher priority.

The CC restricts its access to the regions depending on the current state of the message buffer. The application must adhere to these restrictions in order to ensure data consistency. The receive message buffer states are given in the figure below. A description of the message buffer states is given in Table 47-40, which also provides the access scheme for the access regions.

The status bits FR_MBCCSRn[EDS] and FR_MBCCSRn[LCKS] provide the application with the required status information. The internal status information is not visible to the application.

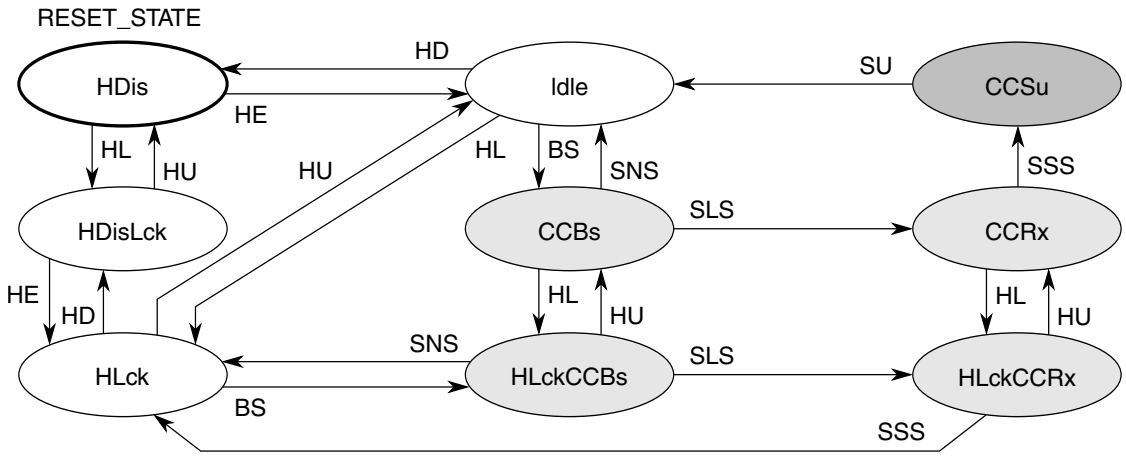


Figure 47-17. Receive Message Buffer States

Table 47-45. Receive Message Buffer States and Access (Sheet 2 of 2)

State	FR_MBCCSRn		Access from		Description
	EDS	LCKS	Appl.	Module	
Idle	1	0	–	SR	**Idle** - Message Buffer is idle. Included in message buffer search.
HDis	0	0	CFG	–	**Disabled** - Message Buffer under configuration. Excluded from message buffer search.
HDisLck	0	1	CFG	–	**Disabled and Locked** - Message Buffer under configuration.Excluded from message buffer search.
HLck	1	1	MSG	–	**Locked** - Applications access to data, control, and status.Included in message buffer search.
CCBs	1	0	–	–	**Buffer Subscribed** - Message buffer subscribed for reception. Filter matches next (slot, cycle, channel) tuple.
HLckCCBs	1	1	MSG	–	**Locked and Buffer Subscribed** - Applications access to data, control, and status. Message buffer subscribed for reception.
CCRx	1	0	–	–	**Message Receive** - Message data received into related shadow buffer.
HLckCCRx	1	1	MSG	–	**Locked and Message Receive** - Applications access to data, control, and status. Message data received into related shadow buffer.
CCSu	1	0	–	RX	**Status Update** - Message buffer status update. Update of status flags, the slot status field, and the header index.

47.8.6.3.1 Message Buffer Transitions

Application transitions, message buffer enable and disable, message buffer lock and unlock, module transitions, and transition priorities are discussed in this section.

47.8.6.3.1.1 Application Transitions

The application transitions that can be triggered by the application using the commands described in Table 47-41. The application issues the commands by writing to the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn). Only one command can be issued with one write access. Each command is executed immediately. If the command is ignored, it must be issued again.

47.8.6.3.1.2 Message Buffer Enable and Disable

The enable and disable commands issued by writing 1 to the trigger bit FR_MBCCSRn[EDT]. The transition that will be triggered by each of these command depends on the current value of the status bit FR_MBCCSRn[EDS]. If the command triggers the disable transition HD and the message buffer is in one of the states CCBs, HLckCCBs, or CCRx, the disable transition has no effect (command is ignored) and the message buffer state is not changed. No notification is given to the application.

47.8.6.3.1.3 Message Buffer Lock and Unlock

The lock and unlock commands issued by writing 1 to the trigger bit FR_MBCCSRn[LCKT]. The transition that will be triggered by each of these commands depends on the current value of the status bit FR_MBCCSRn[LCKS]. If the command triggers the lock transition HL while the message buffer is in the state CCRx, the lock transition has no effect (command is ignored) and message buffer state is not changed. In this case, the message buffer lock error flag LCK_EF in the CHI Error Flag Register (FR_CHIERFR) is set.

Table 47-46. Receive Message Buffer Application Transitions

Transition	Host Command	Condition	Description
HE	FR_MBCCSRn[EDT]:= 1	FR_MBCCSRn[EDS] = 0	Application triggers message buffer enable.
HD		FR_MBCCSRn[EDS] = 1	Application triggers message buffer disable.
HL	FR_MBCCSRn[LCKT]: = 1	FR_MBCCSRn[LCKS] = 0	Application triggers message buffer lock.
HU		FR_MBCCSRn[LCKS] = 1	Application triggers message buffer unlock.

47.8.6.3.1.4 Module Transitions

The module transitions that can be triggered by the CC are described in the next table. Each transition will be triggered for certain message buffers when the related condition is fulfilled.

Table 47-47. Receive Message Buffer Module Transitions

Transition	Condition	Description
BS	slot match and CycleCounter match	**Buffer Subscribed** - The message buffer filter matches next slot and cycle.
SLS	slot start	**Slot Start** - Start of either Static Slot or Dynamic Slot.
SNS	symbol window start or NIT start	**Symbol Window or NIT Start** - Start of either Symbol Window or NIT.
SSS	slot start or symbol window start or NIT start	**Slot or Segment Start** - Start of either Static Slot, Dynamic Slot, Symbol Window, or NIT.
SU	status updated	**Status Updated** - Slot Status field, message buffer status flags, header index updated. Interrupt flag set.

47.8.6.3.1.5 Transition Priorities

The application can trigger only one transition at a time. There is no need to specify priorities among them.

As shown in the table below, the module transitions have a higher priority than the application transitions. For all states except the CCRx state, a module transition and the application lock/unlock transition HL/HU and can be executed at the same time. The result state is reached by first applying the module transition and subsequently the application transition to the intermediately reached state. For example, if the message buffer is in the buffer subscribed state CCBs and the module triggers the slot start transition SLS at the same time as the application locks the message buffer by the HL transition, the intermediate state is CCRx and the resulting state is locked buffer subscribed state HLckCCRx.

Table 47-48. Receive Message Buffer Transition Priorities

State	Priorities	Description
module vs. application		
Idle	BS > HD	Buffer Subscribed > Message Buffer Disable
HLck	BS > HD	Buffer Subscribed > Message Buffer Disable
CCRx	SSS > HL	Slot or Segment Start > Message Buffer Lock

47.8.6.3.2 Message Reception

As a result of the message buffer search, the CC changes the state of up to two enabled receive message buffers from either idle state Idle or locked state HLck to the either subscribed state CCBs or locked buffer subscribed state HLckCCBs by triggering the buffer subscribed transition BS.

If the receive message buffers for the next slot are assigned to both channels, then at most one receive message buffer is changed to a buffer subscribed state.

If more than one matching message buffers assigned to a certain channel, then only the message buffer with the lowest message buffer number is in one of the states mentioned above.

With the start of the next static or dynamic slot the module trigger the slot start transition SLS. This changes the state of the subscribed receive message buffers from either CCBs to CCRx or from HLckCCBs to HLckCCRx, respectively.

During the reception slot, the received frame data are written into the shadow buffers. For details on receive shadow buffers, see Receive Shadow Buffers Concept. The data and status of the receive message buffers that are the CCRx or HLckCCRx are not modified in the reception slot.

47.8.6.3.3 Message Buffer Update

With the start of the next static or dynamic slot or with the start of the symbol window or NIT, the module triggers the slot or segment start transition SSS. This transition changes the state of the receiving receive message buffers from either CCRx to CCSu or from HLckCCRx to HLck, respectively.

If a message buffer was in the locked state HLckCCRx, no update will be performed. The received data are lost. This is indicated by setting the Frame Lost Channel A/B Error Flag FRLA_EF/FRLB_EF in the CHI Error Flag Register (FR_CHIERFR).

If a message buffer was in the CCRx state it is now in the CCSu state. After the evaluation of the slot status provided by the PE the message buffer is updated. The message buffer update depends on the slot status bits and the segment the message buffer is assigned to. This is described in Table 47-51.

Table 47-49. Receive Message Buffer Update (Continued)

vSS!ValidFrame	*vRF!Header! NFIndicator*	Update description
1	1	**Valid non-null frame received.** - Message Buffer Data Field updated. - Frame Header Field updated. - Slot Status Field updated. - DUP:= 1 - DVAL:= 1 - MBIF:= 1
1	0	**Valid null frame received.**

Table continues on the next page...

Table 47-49. Receive Message Buffer Update (Continued) (continued)

vSS!ValidFrame	*vRF!Header! NFIndicator*	Update description
		- Message Buffer Data Field *not* updated. - Frame Header Field *not* updated. - Slot Status Field updated. - DUP:= 0 - DVAL *not* changed - MBIF:= 1
0	x	**No valid frame received.** - Message Buffer Data Field not updated. - Frame Header Field not updated. - Slot Status Field updated. - DUP:= 0 - DVAL *not* changed. - MBIF:= 1, if the slot was not an empty dynamic slot. **Note:** An empty dynamic slot is indicated by the following frame and slot status bit values: *vSS!ValidFrame* = 0 and *vSS!SyntaxError* = 0 and *vSS! ContentError* = 0 and *vSS!BViolation* = 0.

Note

If the number of the last slot in the current communication cycle on a given channel is n, then all receive message buffers assigned to this channel with FR_MBFIDRn[FID] > *n* will not be updated at all.

When the receive message buffer update has finished the status updated transition SU is triggered, which changes the buffer state from CCSu to Idle. An example receive message buffer timing and state change diagram for a normal frame reception is given in the following figure.

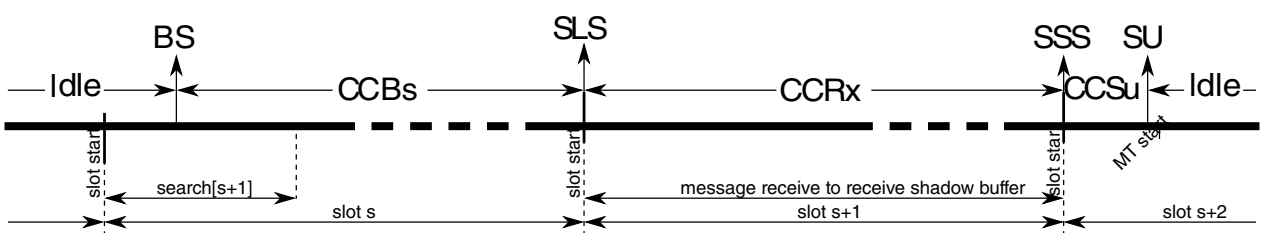


Figure 47-18. Message Reception Timing

The amount of message data written into the message buffer data field of the receive shadow buffer is determined by the following two items:

1. the message buffer segment that the message buffer is assigned to, as defined by the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR).
2. the message buffer data field size, as defined by the related field of the Message Buffer Data Size Register (FR_MBDSR)
3. the number of bytes received over the FlexRay bus

If the message buffer is assigned to the message buffer segment 1, and the number of received bytes is greater than 2 ×FR_MBDSR.MBSEG1DS, the CC writes only 2 × FR_MBDSR.MBSEG1DS bytes into the message buffer data field of the receive shadow buffer. If the number of received bytes is less than 2 × FR_MBDSR.MBSEG1DS, the CC writes only the received number of bytes and will not change the trailing bytes in the message buffer data field of the receive shadow buffer. The same holds for the message buffer segment 2 with FR_MBDSR.MBSEG2DS.

47.8.6.3.4 Received Message Access

To access the message data received over the FlexRay bus, the application reads the message data stored in the message buffer data field of the corresponding receive message buffer. The access to the message buffer data field is described in Individual Message Buffers.

The application can read the message buffer data field if the receive message buffer is one of the states HDis, HDisLck, or HLck. If the message buffer is in one of these states, the CC will not change the content of the message buffer.

47.8.6.3.5 Receive Shadow Buffers Concept

The receive shadow buffer concept applies only to individual receive message buffers. The intention of this concept is to ensure that only syntactically and semantically valid received non-null frames are presented to the application in a receive message buffer. The basic structure of a receive shadow buffer is described in Receive Shadow Buffers.

The receive shadow buffers temporarily store the received frame header and message data. After the slot boundary the slot status information is generated. If the slot status information indicates the reception of the valid non-null frame (see Table 47-49), the CC writes the slot status into the slot status field of the receive shadow buffer and exchanges the content of the Message Buffer Index Registers (FR_MBIDXRn) with the content of the corresponding internal shadow buffer index register. In all other cases, the CC writes the slot status into the identified receive message buffer, depending on the slot status and the FlexRay segment the message buffer is assigned to.

The shadow buffer concept, with its index exchange, results in the fact that the FlexRay memory area located message buffer associated to an individual receive message buffer changes after successful reception of a valid frame. This means that the message buffer area in the FlexRay memory area accessed by the application for reading the received message is different from the initial setting of the message buffer. Therefore, the application must not rely on the index information written initially into the Message Buffer Index Registers (FR_MBIDXRn). Instead, the index of the message buffer header field must be fetched from the Message Buffer Index Registers (FR_MBIDXRn).

47.8.7 Individual Message Buffer Search

This section provides a detailed description of the message buffer search algorithm.

The message buffer search determines for each enabled channel if a slot s in a communication cycle c is assigned for frame or null frame transmission or if it is subscribed for frame reception on that channel.

The message buffer search is a sequential algorithm which is invoked at the following protocol related events:

1. NIT start
2. slot start in the static segment
3. minislot start in the dynamic segment

The message buffer search within the NIT searches for message buffers assigned or subscribed to slot 1. The message buffer search within slot *n* searches for message buffers assigned or subscribed to slot *n+1*.

In general, the message buffer search for the next slot *n* considers only message buffers which are

1. enabled, i.e. FR_MBCCSRn[EDS] = 1, and
2. matches the next slot *n*, i.e. FR_MBFIDRn[FID] = *n*, and

On top of that, for the static segment only those message buffers are considered, that match the condition of at least one row of the "Message Buffer Search Priority (static segment)" table shown below. For the dynamic segment only those message buffers are considered, that match the condition of at least one row of the next "Message Buffer Search Priority (dynamic segment)" table. These message buffers are called *matching* message buffers.

For each enabled channel the message buffer search may identify multiple *matching* message buffers. Among all matching message buffers the message buffers with highest priority according to the "Message Buffer Search Priority (static segment)" table for the static segment and according to the "Message Buffer Search Priority (dynamic segment)" table for the dynamic segment are selected.

Table 47-50. Message Buffer Search Priority (static segment)

Priority	MTD	LCKS	CMT	CCFM[1]	Description	Transition
(highest) 0	1	0	1	1	transmit buffer, matches cycle count, not locked and committed	MA
1	1	-	0	1	transmit buffer, matches cycle count, not committed	SA
	1	1	-	1	transmit buffer, matches cycle count, locked	SA
2	1	-	-	-	transmit buffer	SA
3	0	0	n/a	1	receive buffer, matches cycle count, not locked	SB
(lowest) 4	0	1	n/a	1	receive buffer, matches cycle count, locked	SB

1. Cycle Counter Filter Match, see Message Buffer Cycle Counter Filtering.

Table 47-51. Message Buffer Search Priority (dynamic segment)

Priority	MTD	LCKS	CMT	CCFM[1]	Description	Transition
(highest) 0	1	0	1	1	transmit buffer, matches cycle count, not locked and committed	MA
1	0	0	n/a	1	receive buffer, matches cycle count, not locked	SB
(lowest) 2	0	1	n/a	1	receive buffer, matches cycle count, locked	SB

1. Cycle Counter Filter Match, see Message Buffer Cycle Counter Filtering.

If there are multiple message buffer with highest priority, the message buffer with the lowest message buffer number is selected. All message buffer which have the highest priority must have a consistent channel assignment as specified in Message Buffer Channel Assignment Consistency.

Depending on the message buffer channel assignment the same message buffer can be found for both channel A and channel B. In this case, this message buffer is used as described in Individual Message Buffers.

47.8.7.1 Message Buffer Cycle Counter Filtering

The message buffer cycle counter filter is a value-mask filter defined by the CCFE, CCFMSK, and CCFVAL fields in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn). This filter determines a set of communication cycles in which the

message buffer is considered for message reception or message transmission. If the cycle counter filter is disabled, i.e. CCFE = 0, this set of cycles consists of all communication cycles.

If the cycle counter filter of a message buffer does not match a certain communication cycle number, this message buffer is not considered for message transmission or reception in that communication cycle. In case of a transmit message buffer assigned to a slot in the static segment, though, this buffer is added to the matching message buffers to indicate the slot assignment and to trigger the null frame transmission.

The cycle counter filter of a message buffer matches the communication cycle with the number CYCCNT if at least one of the following conditions evaluates to true:

$$MBCCFRn[CCFE] = 0$$

$$CYCCNT \,\&\, MBCCFRn[CCFMSK] = MBCCFRn[CCFVAL] \,\&\, MBCCFRn[CCFMSK]$$

47.8.7.2 Message Buffer Channel Assignment Consistency

The message buffer channel assignment given by the CHA and CHB bits in the Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn) defines the channels on which the message buffer will receive or transmit. The message buffer with number *n* transmits or receives on channel A if FR_MBCCFRn[CHA] = 1 and transmits or receives on channel B if FR_MBCCFRn[CHB] = 1.

To ensure correct message buffer operation, all message buffers assigned to the same slot and with the same priority must have a *consistent* channel assignment. That means they must be either assigned to one channel only, or must be assigned to *both* channels. The behavior of the message buffer search is not defined, if both types of channel assignments occur for one slot and priority. An inconsistent channel assignment for message buffer 0 and message buffer 1 is depicted in the figure below.

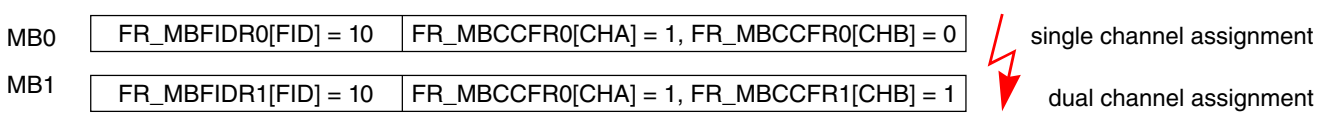


Figure 47-19. Inconsistent Channel Assignment

47.8.7.3 Node Related Slot Multiplexing

The term *Node Related Slot Multiplexing* applies to the dynamic segment only and refers to the functionality if there are transmit as well as receive message buffers are configured for the same slot.

According to Table 47-51 the transmit buffer is only found if the cycle counter filter matches, and the buffer is not locked and committed. In all other cases, the receive buffer will be found. Thus, if the block has no data to transmit in a dynamic slot, it is able to receive frames on that slot.

47.8.7.4 Message Buffer Search Error

There are two kinds of errors which may occur during message buffer search[1].

47.8.7.4.1 Message Buffer Search Start while Running

If the message buffer search is running in slot *n-1* and the next message buffer search start event appears due to the start of slot *n*, the message buffer search engine is stopped and the Message Buffer Search Error Flag MBS_EF is set in the CHI Error Flag Register (FR_CHIERFR). As a result of this stop, no individual message buffer is identified for transmission or reception in slot *n*. Additionally, the search engine will not be started in slot *n*, and consequently no individual message buffer is identified for transmission or reception in slot *n+1*.

A message buffer search error appears only if the CHI frequency is too slow to allow the search through all message buffers to be completed within the NIT or a minislot.

For more details of minimum required CHI frequency see Number of Usable Message Buffers.

47.8.7.4.2 Illegal Message Buffer Index Found

If the message buffer search has finished the message buffer search in slot *n-1*, it retrieves the data offset values for the found message buffers and the receive shadow buffers. If one of these message buffers contains an illegal message buffer index, the Message Buffer Search Error Flag MBS_EF is set in the CHI Error Flag Register (FR_CHIERFR) is set and no individual message buffer is identified for transmission or reception in slot *n*. The legal message buffer index values for the individual and receive shadow buffers are specified in the Receive Shadow Buffer Index Register section (FR_RSBIR) and the Message Buffer Index Registers sections (FR_MBIDXRn).

1. The FIFO reception is not affected by the search errors. Additionally, if no rx buffer has been found due to an search error, the received frame is considered for FIFO reception.

47.8.8 Individual Message Buffer Reconfiguration

The initial configuration of each individual message buffer can be changed even when the protocol is not in the *POC:config* state. This is referred to as individual message buffer *reconfiguration*. The configuration bits and fields that can be changed are given in the section on Specific Configuration Data. The common configuration data given in the section on Specific Configuration Data can not be reconfigured when the protocol is out of the *POC:config* state.

47.8.8.1 Reconfiguration Schemes

Depending on the target and destination basic state of the message buffer that is to be reconfigured, there are three reconfiguration schemes.

47.8.8.1.1 Basic Type Not Changed (RC1)

A reconfiguration will not change the basic type of the individual message buffer, if the message buffer transfer direction bit FR_MBCCSRn[MTD] are not changed. This type of reconfiguration is denoted by RC1 in Figure 47-20. Transmit and receive message buffers can be RC1-reconfigured when in the HDis or HDisLck state.

47.8.8.1.2 Buffer Type Not Changed (RC2)

A reconfiguration will not change the buffer type of the individual message buffer. This type of reconfiguration is denoted by RC2 in the figure below. It applies to transmit and receive message buffers. Transmit and receive message buffers can be RC2-reconfigured when in the HDis or HDisLck state.

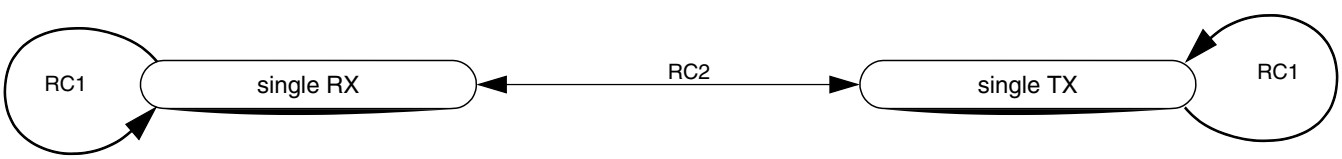


Figure 47-20. Message Buffer Reconfiguration Scheme

47.8.9 Receive FIFOs

This section provides the functional description of the two receive FIFOs.

47.8.9.1 Overview

The two receive FIFOs implement the queued message buffer concept defined by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*. One FIFO is assigned to channel A, the other FIFO is assigned to channel B. Both FIFOs work completely independent from each other.

The message buffer structure of each FIFO is described in Receive FIFO. The area in the FlexRay memory area for each of the two FIFOs is characterized by:

- *The FIFO system memory base address*
- *The index of the first FIFO entry given by Receive FIFO Start Index Register (FR_RFSIR)*
- *The data field offset of the data field belonging to the first FIFO entry given by Receive FIFO Start Data Offset Register (FR_RFSDOR)*
- *The number of FIFO entries and the length of each FIFO entry as given by Receive FIFO Depth and Size Register (FR_RFDSR)*

47.8.9.2 FIFO Configuration

The FIFOs can be configured for two different locations of the system memory base address via the FIFO address mode bit FAM in the Module Configuration Register (FR_MCR).

47.8.9.2.1 Single System Memory Base Address Mode

This mode is configured, when the FIFO address mode flag FR_MCR[FAM] is set to 0. In this mode, the location of the system memory base address for the FIFO buffers is System Memory Base Address Register (FR_SYMBADR).

47.8.9.2.2 Dual System Memory Base Address Mode

This mode is configured, when the FIFO address mode flag FR_MCR[FAM] is set to 1. In this mode, the location of the system memory base address for the FIFO buffers is Receive FIFO System Memory Base Address Register (FR_RFSYMBADR).

The FIFO control and configuration data are given in Receive FIFO Control and Configuration Data. The configuration of the FIFOs consists of two steps.

The first step is the allocation of the required amount of FlexRay memory area for the FlexRay window. This includes the allocation of the message buffer header area and the allocation of the message buffer data fields. For more details see FlexRay Memory Area Layout.

The second step is the programming of the configuration data register while the PE is in *POC:config*.

The following steps configure the layout of the FIFO.

- Configure the FIFO update and address modes in Module Configuration Register (FR_MCR)
- Configure the FIFO system memory base address
- Configure the Receive FIFO Start Index Register (FR_RFSIR) with the first message buffer header index that belongs to the FIFO
- Configure the Receive FIFO Start Data Offset Register (FR_RFSDOR) with the data field offset of the data field belonging to the first message buffer that belongs to the FIFO
- Configure the Receive FIFO Depth and Size Register (FR_RFDSR) with FIFO entry size
- Configure the Receive FIFO Depth and Size Register (FR_RFDSR) with FIFO depth
- Configure the FIFO Filters

47.8.9.3 FIFO Periodic Timer

The FIFO periodic timer is used to generate an FIFO almost-full interrupt at certain point in time, if the almost-full watermark is not reached, but the FIFO is not empty. This can be used to prevent frames from get stuck in the FIFO for a long time.

The FIFO periodic timer is configured via the Receive FIFO Periodic Timer Register (FR_RFPTR). If the periodic timer duration FR_RFPTR[PTD] is configured to 0x0000, the periodic timer is continuously expired. If the periodic timer duration FR_RFPTR[PTD] is configured to 0x3FFF, the periodic timer never expires. If the periodic timer is configured to a value *ptd*, greater than 0x0000 and smaller 0x3FFF, the periodic timer expires and is restarted at the start of every communication cycle, and expires and is restarted after *ptd* macroticks have been elapsed.

47.8.9.4 FIFO Reception

The FIFO reception is a CC internal operation.

A message frame reception is directed into the FIFO, if no individual message buffer is assigned for transmission or subscribed for reception for the current slot. In this case the FIFO filter path shown in Figure 47-21 is activated.

If the FIFO filter path indicates that the received frame has to be appended to the FIFO and the FIFO is not full, the CC writes the received frame header into the message buffer header field indicated by the CC internal FIFO write index. The frame payload data are written into the corresponding message buffer data field. If the status of the received frame indicates a valid non-null frame, the slot status information is written into the message buffer header field and the CC internal FIFO write index is updated by 1 and the fifo fill level FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is incremented. If the status of the received frame indicates an invalid or null frame, the frame is not appended to the FIFO.

47.8.9.5 FIFO Almost-Full Interrupt Generation

If the fifo fill level FLA (FLB) is updated after a frame reception and exceeds the FIFO watermark level WM, i.e. FLA>WM_A (FLB>WM_B), then the FIFO almost-full interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is asserted.

If the periodic timer expires, and FIFOA (FIFOB) is not empty, i.e. FLA>0 (FLB>0), then the FIFO almost-full interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is asserted.

47.8.9.6 FIFO Overflow Error Generation

If the FIFOA (FIFOB) is full, i.e. FLA=$FIFO_DEPTH_A$ (FLB=$FIFO_DEPTH_B$) and the conditions for a FIFO reception as described in FIFO Reception are fulfilled, then the fifo overflow error flag FR_CHIERFR[FOVA_EF] (FR_CHIERFR[FOVB_EF]) is asserted.

47.8.9.7 FIFO Message Access

The FIFOA (FIFOB) contains valid messages if the FIFO fill level given in the fields FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is greater than 0. The Receive FIFO A Read Index Register (FR_RFARIR) and the

(Receive FIFO B Read Index Register (FR_RFBRIR)) point to a message buffer with valid content and the oldest frames stored in the FIFO.The respective read data field offsets can be calculated according to Equation 1-125.

If the FIFO fill level FLA (FLB) in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR) is 0, than the FIFOA (FIFOB) contains no valid messages and the corresponding read index register Receive FIFO A Read Index Register (FR_RFARIR) or (Receive FIFO B Read Index Register (FR_RFBRIR) point to a message buffer with invalid content. In this case the application must not read data from this FIFO.

To access the oldest message in the FIFOA (FIFOB), the application first reads the read index RDIDX out of the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)). This read index points to the message buffer header field of the oldest message buffer that contains valid received message data. The data field offset belonging to this message buffer must be calculated by the application according to Equation 1-125. The application can access the message data as described in Receive FIFO. When the application has read the message buffer data and status information, it can update the FIFO as described in FIFO Update.

47.8.9.8 FIFO Update

The application updates the FIFOA (FIFOB) by writing a pop count value *pc* different from 0 to the PCA (PCB) field in the Receive FIFO Fill Level and POP Count Register (FR_RFFLPCR).

As a result of the this operation, the CC removes the oldest *pc* entries from FIFOA (FIFOB).

If the specified pop count value *pc* is greater than the current fill level *fl* provided in FLA (FAB) field, then only *fl* entries are removed from the FIFOA (FIFOB), the remaining *fl-pc* requested pop operations are discarded without any notification. In this case FIFOA (FIFOB) is empty after the update operation.

The read index in the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)) is incremented by the number of removed items. If the read index reaches the top of the FIFO, it wraps around to the FIFO start index defined in Receive FIFO Start Index Register (FR_RFSIR) automatically.

47.8.9.8.1 FIFO Interrupt Flag Update

The FIFO Interrupt Flag Update mode is configured, when the FIFO update mode flag FR_MCR[FUM] is set to 0. In this mode FIFOA (FIFOB) will be updated by 1 entry, when the interrupt flag FR_GIFER[FAFAIF] (FR_GIFER[FAFBIF]) is written with 1 by the application.

If the FIFO is empty, the update request is ignored without any notification.

The read index in the Receive FIFO A Read Index Register (FR_RFARIR) (Receive FIFO B Read Index Register (FR_RFBRIR)) is incremented by 1, if the FIFO was not empty. If the read index reaches the top of the FIFO, it wraps around to the FIFO start index automatically.

47.8.9.9 FIFO Filtering

The FIFO filtering is activated after all enabled individual receive message buffers have been searched without success for a message buffer to receive the current frame.

The CC provides three sets of FIFO filters. The FIFO filters are applied to valid non-null frames only. The FIFO will not receive invalid or null-frames. For each FIFO filter, the pass criteria is specified in the related section given below. Only frames that have passed all filters will be appended to the FIFO. The FIFO filter path is depicted in the figure below.

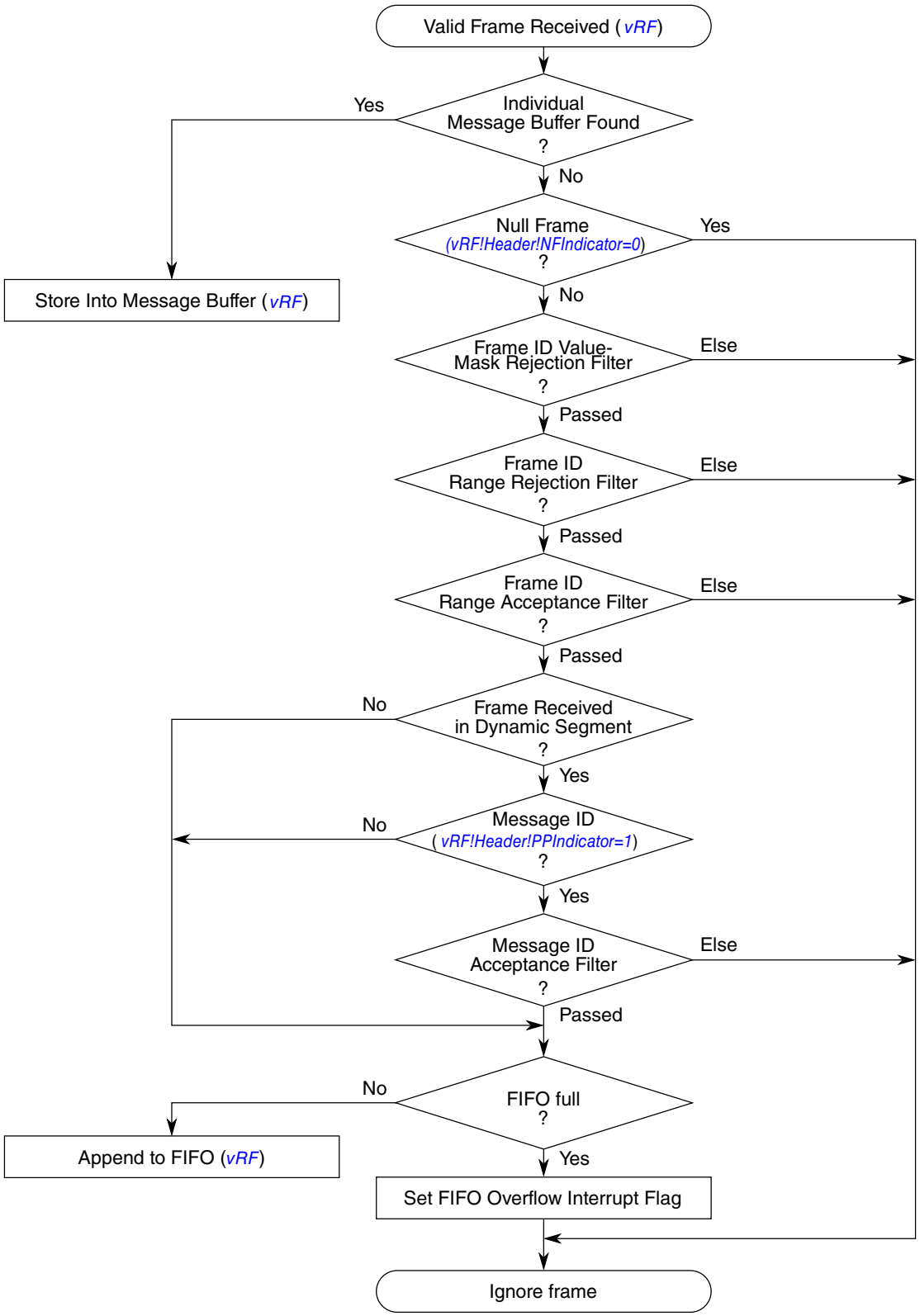


Figure 47-21. Received Frame FIFO Filter Path

A received frame passes the FIFO filtering if it has passed all three type of filter.

47.8.9.9.1 RX FIFO Frame ID Value-Mask Rejection Filter

The frame ID value-mask rejection filter is a value-mask filter and is defined by the fields in the Receive FIFO Frame ID Rejection Filter Value Register (FR_RFFIDRFVR) and the Receive FIFO Frame ID Rejection Filter Mask Register (FR_RFFIDRFMR). Each received frame with a frame ID FID that does not match the value-mask filter value passes the filter, i.e. is not rejected.

Consequently, a received valid frame with the frame ID FID passes the RX FIFO Frame ID Value-Mask Rejection Filter if Equation 21 on page 2034 is fulfilled.

$$FID \,\&\, FR_RFFIDRFMR[FIDRFMSK] \neq FR_RFFIDRFVR[FIDRFVAL] \,\&\, FR_RFFIDRFMR[FIDRFMSK]$$

Equation 21

The RX FIFO Frame ID Value-Mask Rejection Filter can be configured to pass all frames by the following settings.

$$FR_RFFIDRFVR[FIDRFVAL] = 0x000 \text{ and } FR_RFFIDRFMR[FIDRFMSK] = 0x7FF$$

Equation 22

Using the settings above, only the frame with frame ID 0 will be rejected, which is an invalid frame. All other frames will pass.

The RX FIFO Frame ID Value-Mask Rejection Filter can be configured to reject all frames by the following settings.

$$FR_RFFIDRFMR[FIDRFMSK] = 0x000$$

Equation 23

Using these settings, Equation 21 on page 2034 can never be fulfilled (0!= 0) and thus all frames are rejected; no frame will pass. This is the reset value for the RX FIFO.

47.8.9.9.2 RX FIFO Frame ID Range Rejection Filter

Each of the four RX FIFO Frame ID Range filters can be configured as a rejection filter. The filters are configured by the Receive FIFO Range Filter Configuration Register (FR_RFRFCFR) and controlled by the Receive FIFO Range Filter Control Register (FR_RFRFCTR). The RX FIFO Frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range rejection filters if either no rejection filter is enabled, or, for all of the enabled RX FIFO Frame ID Range rejection filters, i.e. FR_RFRFCTR[FiMD] = 1 and FR_RFRFCTR[FiEN] = 1, Equation 24 on page 2035 is fulfilled.

$$FID < FR_RFRFCFR_{SEL}[SID_{IBD=0}]) or (FR_RFRFCFR_{SEL}[SID_{IBD=1}] < FID)$$

Equation 24

Consequently, all frames with a frame ID that fulfills Equation 25 on page 2035 for at least one of the enabled rejection filters will be rejected and thus not pass.

$$FR_RFRFCFR_{SEL}[SID_{IBD=0}] \leq FID \leq FR_RFRFCFR_{SEL}[SID_{IBD=1}]$$

Equation 25

47.8.9.9.3 RX FIFO Frame ID Range Acceptance filter

Each of the four RX FIFO Frame ID Range filters can be configured as an acceptance filter. The filters are configured by the Receive FIFO Range Filter Configuration Register (FR_RFRFCFR) and controlled by the Receive FIFO Range Filter Control Register (FR_RFRFCTR). The RX FIFO Frame ID range filters apply to all received valid frames. A received frame with the frame ID FID passes the RX FIFO Frame ID Range acceptance filters if either no acceptance filter is enabled, or, for at least one of the enabled RX FIFO Frame ID Range acceptance filters, i.e. FR_RFRFCTR[FiMD] = 0 and FR_RFRFCTR[FiEN] = 1, Equation 26 on page 2035 is fulfilled.

$$FR_RFRFCFR_{SEL}[SID_{IBD=0}] \leq FID \leq FR_RFRFCFR_{SEL}[SID_{IBD=1}]$$

Equation 26

47.8.9.9.4 RX FIFO Message ID Acceptance Filter

The RX FIFO Message ID Acceptance Filter is a value-mask filter and is defined by the Receive FIFO Message ID Acceptance Filter Value Register (FR_RFMIDAFVR) and the Receive FIFO Message ID Acceptance Filter Mask Register (FR_RFMIDAFMR). This filter applies only to valid frames received in the dynamic segment with the payload preamble indicator bit PPI set to 1. All other frames will pass this filter.

A received valid frame in the dynamic segment with the payload preamble indicator bit PPI set to 1 and with the message ID MID (the first two bytes of the payload) will pass the RX FIFO Message ID Acceptance Filter if Equation 27 on page 2035 is fulfilled.

$$MID \,\&\, FR_RFMIDAFMR[MIDAFMSK] = FR_RFMIDAFMR[MIDAFVAL] \,\&\, FR_RFMIDAFMR[MIDAFMSK]$$

Equation 27

The RX FIFO Message ID Acceptance Filter can be configured to accept all frames by setting

$$FR_RFMIDAFMR[MIDAFMSK] := 0x000$$

Equation 28

Using these settings, Equation 27 on page 2035 is always fulfilled and all frames will pass.

47.8.10 Channel Device Modes

This section describes the two FlexRay channel device modes that are supported by the CC.

47.8.10.1 Dual Channel Device Mode

In the dual channel device mode, both FlexRay ports are connected to physical FlexRay bus lines. The FlexRay port consisting of FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ is connected to the physical bus channel A and the FlexRay port consisting of FR_B_RX, FR_B_TX, and $\overline{\text{FR_B_TX_EN}}$ is connected to the physical bus channel B. The dual channel system is shown in the following figure.

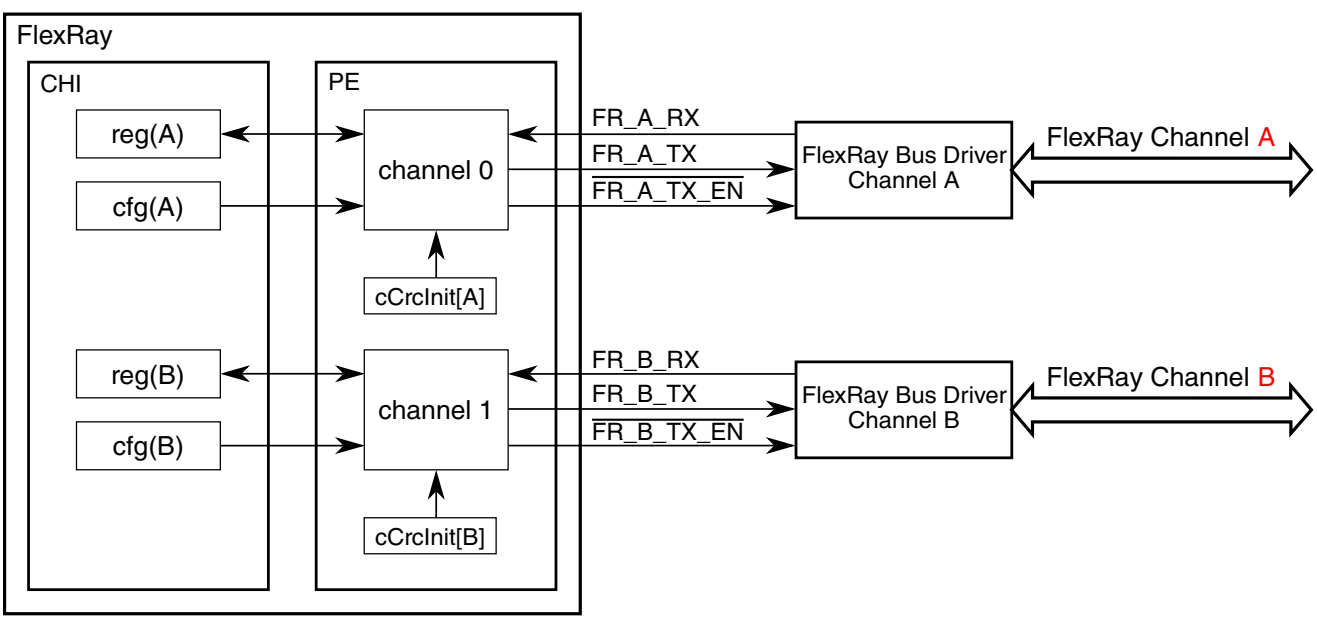


Figure 47-22. Dual Channel Device Mode

47.8.10.2 Single Channel Device Mode

The single channel device mode supports devices that have only one FlexRay port available. This FlexRay port consists of the signals FR_A_RX, FR_A_TX, and $\overline{\text{FR_A_TX_EN}}$ and can be connected to either the physical bus channel A (shown in the "Single Channel Device Mode (Channel A)" figure below) or the physical bus channel B (shown in the "Single Channel Device Mode (Channel B)" figure below).

If the device is configured as a single channel device by setting FR_MCR[SCM] to 1, only the internal channel A and the FlexRay Port A is used. Depending on the setting of FR_MCR[CHA] and FR_MCR[CHB], the internal channel A behaves either as a FlexRay Channel A or FlexRay Channel B. The bit FR_MCR[CHA] must be set, if the FlexRay Port A is connected to a FlexRay Channel A. The bit FR_MCR[CHB] must be set if the FlexRay Port A is connected to a FlexRay Channel B. The two FlexRay channels differ only in the initial value for the frame CRC *cCrcInit*. For a single channel device, the application can access and configure only the registers related to internal channel A.

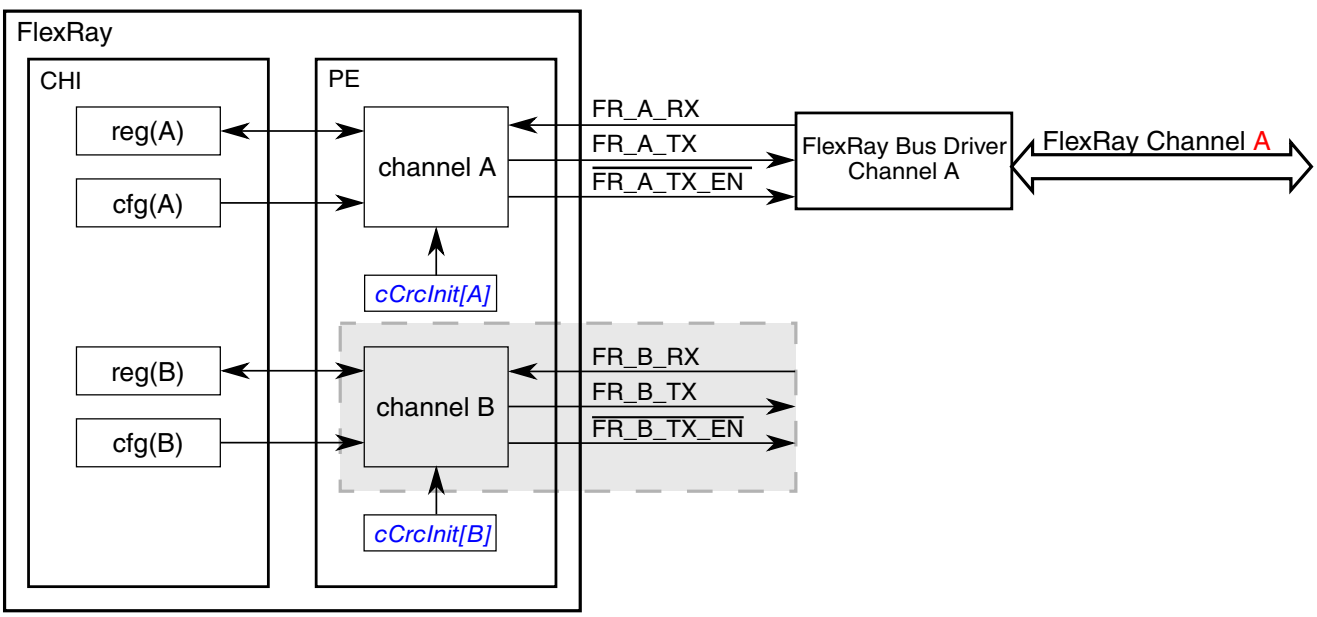


Figure 47-23. Single Channel Device Mode (Channel A)

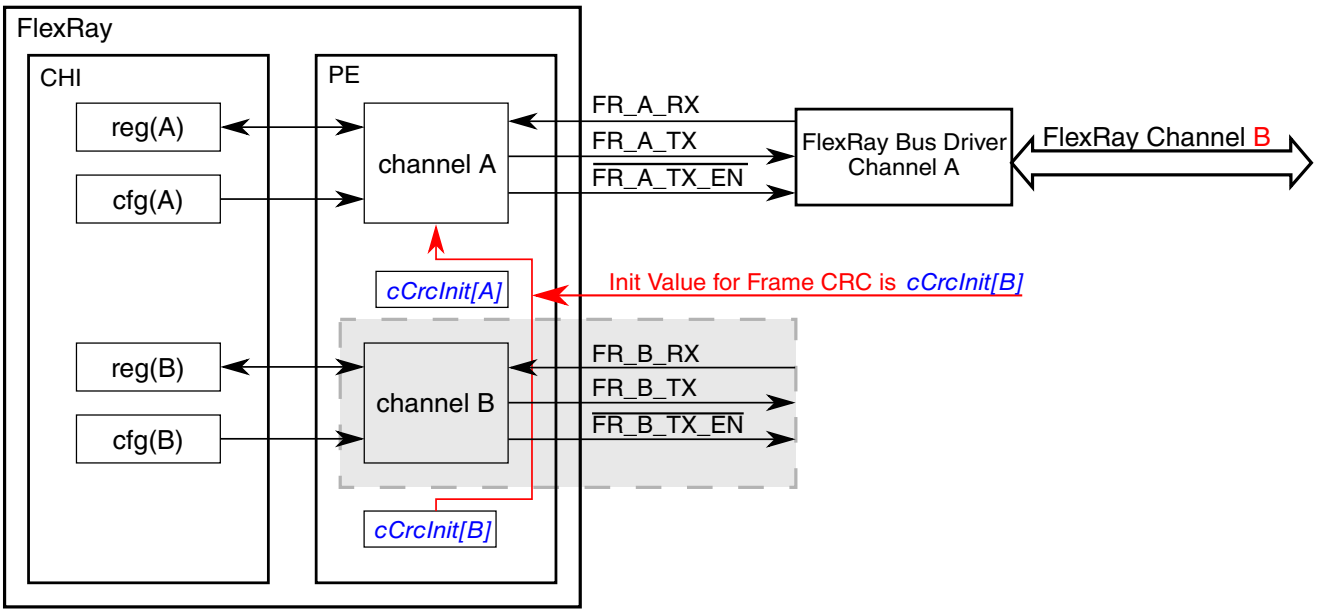


Figure 47-24. Single Channel Device Mode (Channel B)

47.8.11 External Clock Synchronization

The application of the external rate and offset correction is triggered when the application writes to the EOC_AP and ERC_AP fields in the Protocol Operation Control Register (FR_POCR). The PE applies the external correction values in the next even-odd cycle pair as shown in the "External Offset Correction Write and Application Timing" figure and the "External Rate Correction Write and Application Timing" figures below.

Note

The values provided in the EOC_AP and ERC_AP fields are the values that were written from the application most recently. If these value were already applied, they will not be applied in the current cycle pair again.

If the offset correction applied in the NIT of cycle 2n+1 shall be affect by the external offset correction, the EOC_AP field must be written to after the start of cycle 2n and before the end of the static segment of cycle 2n+1. If this field is written to after the end of the static segment of cycle 2n+1, it is not guaranteed that the external correction value is applied in cycle 2n+1. If the value is not applied in cycle 2n+1, then the value will be applied in the cycle 2n+3. Refer to the following for timing details.

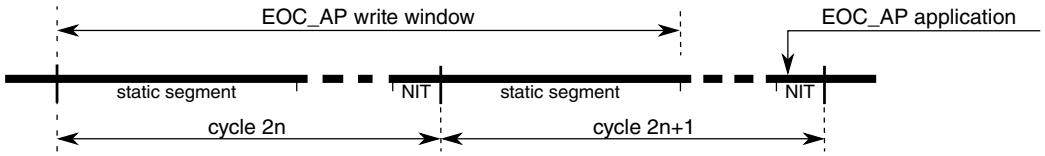


Figure 47-25. External Offset Correction Write and Application Timing

If the rate correction for the cycle pair [2n+2, 2n+3] shall be affect by the external offset correction, the ERC_AP field must be written to after the start of cycle 2n and before the end of the static segment start of cycle 2n+1. If this field is written to after the end of the static segment of cycle 2n+1, it is not guaranteed that the external correction value is applied in cycle pair [2n+2, 2n+3]. If the value is not applied for cycle pair [2n+2, 2n+3], then the value will be applied for cycle pair [2n+4, 2n+5]. Refer to the following for details.

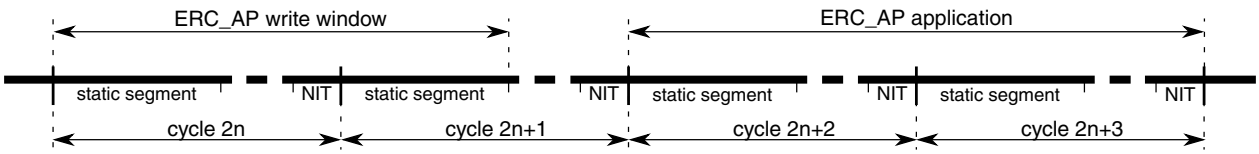


Figure 47-26. External Rate Correction Write and Application Timing

47.8.12 Sync Frame ID and Sync Frame Deviation Tables

The FlexRay protocol requires the provision of a snapshot of the Synchronization Frame ID tables for the even and odd communication cycle for both channels. The CC provides the means to write a copy of these internal tables into the FlexRay memory area and ensures application access to consistent tables by means of table locking. Once the application has locked the table successfully, the CC will not overwrite these tables and the application can read a consistent snapshot.

Note

Only synchronization frames that have passed the synchronization frame filters are considered for clock synchronization and appear in the sync frame tables.

47.8.12.1 Sync Frame ID Table Content

The Sync Frame ID Table is a snapshot of the protocol related variables *vsSyncIdListA* and *vsSyncIdListB* for each even and odd communication cycle. This table provides a list of the frame IDs of the synchronization frames received on the corresponding channel and cycle that are used for the clock synchronization.

47.8.12.2 Sync Frame Deviation Table Content

The Sync Frame Deviation Table is a snapshot of the protocol related variable zsDev(id)(oe)(ch)!Value. Each Sync Frame Deviation Table entry provides the deviation value for the sync frame, with the frame ID presented in the corresponding entry in the Sync Frame ID Table.

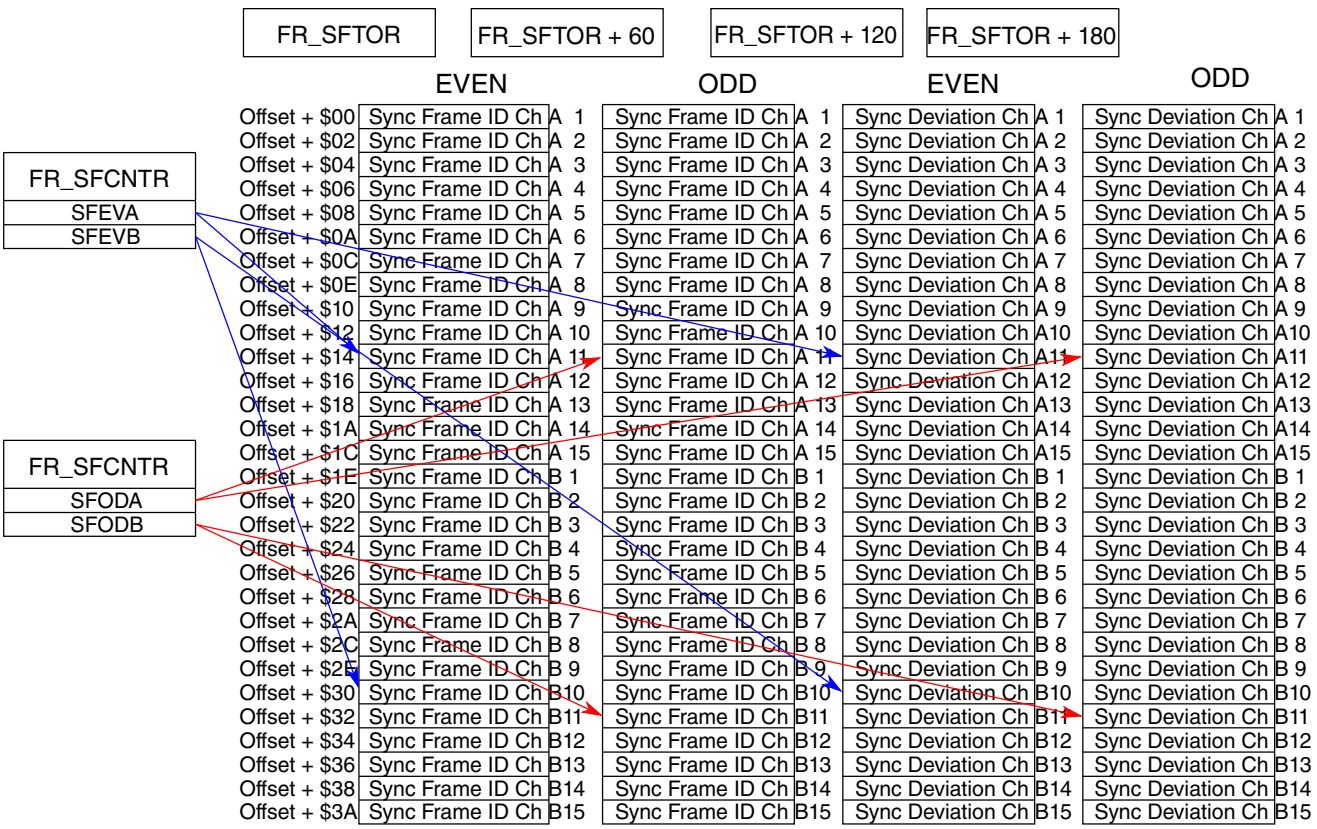


Figure 47-27. Sync Table Memory Layout

47.8.12.3 Sync Frame ID and Sync Frame Deviation Table Setup

The CC writes a copy of the internal synchronization frame ID and deviation tables into the FlexRay memory area if requested by the application. The application must provide the appropriate amount of FlexRay memory area for the tables. The memory layout of the tables is given in Figure 47-27. Each table occupies 120 16-bit entries.

While the protocol is in *POC:config* state, the application must program the offsets for the tables into the Sync Frame Table Offset Register (FR_SFTOR).

47.8.12.4 Sync Frame ID and Sync Frame Deviation Table Generation

The application controls the generation process of the Sync Frame ID and Sync Frame Deviation Tables into the FlexRay memory area using the Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR). A summary of the copy modes is given in the table below.

Table 47-52. Sync Frame Table Generation Modes

FR_SFTCCSR			Description
OPT	SDVEN	SIDEN	
0	0	0	No Sync Frame Table copy
0	0	1	Sync Frame ID Tables will be copied continuously
0	1	0	Reserved
0	1	1	Sync Frame ID Tables and Sync Frame Deviation Tables will be copied continuously
1	0	0	No Sync Frame Table copy
1	0	1	Sync Frame ID Tables for next even-odd-cycle pair will be copied
1	1	0	Reserved
1	1	1	Sync Frame ID Tables and Sync Frame Deviation Tables for next even-odd-cycle pair will be copied

The Sync Frame Table generation process is described in the following for the even cycle. The same sequence applies to the odd cycle.

If the application has enabled the sync frame table generation by setting FR_SFTCCSR[SIDEN] to 1, the CC starts the update of the even cycle related tables after the start of the NIT of the next even cycle. The CC checks if the application has locked the tables by reading the FR_SFTCCSR[ELKS] lock status bit. If this bit is set, the CC will not update the table in this cycle. If this bit is cleared, the CC locks this table and starts the table update. To indicate that these tables are currently updated and may contain inconsistent data, the CC clears the even table valid status bit FR_SFTCCSR[EVAL]. Once all table entries related to the even cycle have been transferred into the FlexRay memory area, the CC sets the even table valid bit FR_SFTCCSR[EVAL] and the Even Cycle Table Written Interrupt Flag EVT_IF in the Protocol Interrupt Flag Register 1 (FR_PIFR1). If the interrupt enable flag EVT_IE is set, an interrupt request is generated.

To read the generated tables, the application must lock the tables to prevent the CC from updating these tables. The locking is initiated by writing a 1 to the even table lock trigger FR_SFTCCSR[ELKT]. When the even table is not currently updated by the CC, the lock is granted and the even table lock status bit FR_SFTCCSR[ELKS] is set. This indicates that the application has successfully locked the even sync tables and the corresponding

status information fields SFRA, SFRB in the Sync Frame Counter Register (FR_SFCNTR). The value in the FR_SFTCCSR[CYCNUM] field provides the number of the cycle that this table is related to.

The number of available table entries per channel is provided in the FR_SFCNTR[SFEVA] and FR_SFCNTR[SFEVB] fields. The application can now start to read the sync table data from the locations given in Figure 47-27.

After reading all the data from the locked tables, the application must unlock the table by writing to the even table lock trigger FR_SFTCCSR[ELKT] again. The even table lock status bit FR_SFTCCSR[ELKS] is reset immediately.

If the sync frame table generation is disabled, the table valid bits FR_SFTCCSR[EVAL] and FR_SFTCCSR[EVAL] are reset when the counter values in the Sync Frame Counter Register (FR_SFCNTR) are updated. This is done because the tables stored in the FlexRay memory area are no longer related to the values in the Sync Frame Counter Register (FR_SFCNTR).

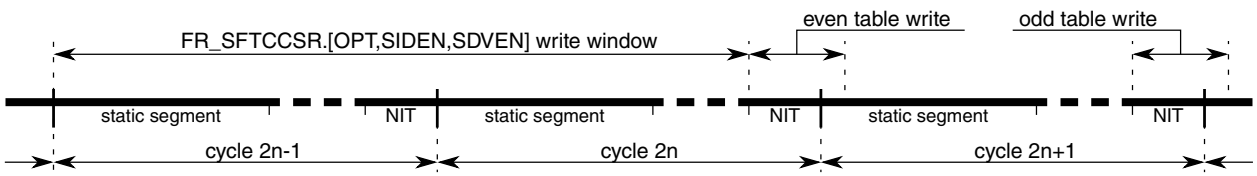


Figure 47-28. Sync Frame Table Trigger and Generation Timing

47.8.12.5 Sync Frame Table Access

The sync frame tables will be transferred into the FlexRay memory area during the table write windows shown in Figure 47-27. During the table write, the application can not lock the table that is currently written. If the application locks the table outside of the table write window, the lock is granted immediately.

47.8.12.5.1 Sync Frame Table Locking and Unlocking

The application locks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT in the Sync Frame Table Configuration, Control, Status Register (FR_SFTCCSR). If the affected table is not currently written to the FlexRay memory area, the lock is granted immediately, and the lock status bit ELKS/OLKS is set. If the affected table is currently written to the FlexRay memory area, the lock is not granted. In this case, the application must issue the lock request again until the lock is granted.

The application unlocks the even/odd sync frame table by writing 1 to the lock trigger bit ELKT/OLKT. The lock status bit ELKS/OLKS is cleared immediately.

47.8.13 MTS Generation

The CC provides a flexible means to request the transmission of the Media Access Test Symbol MTS in the symbol window on channel A or channel B.

The application can configure the set of communication cycles in which the MTS will be transmitted over the FlexRay bus by programming the CYCCNTMSK and CYCCNTVAL fields in the MTS A Configuration Register (FR_MTSACFR) and MTS B Configuration Register (FR_MTSBCFR).

The application enables or disables the generation of the MTS on either channel by setting or clearing the MTE control bit in the MTS A Configuration Register (FR_MTSACFR) or MTS B Configuration Register (FR_MTSBCFR). If an MTS is to be transmitted in a certain communication cycle, the application must set the MTE control bit during the static segment of the preceding communication cycle.

The MTS is transmitted over channel A in the communication cycle with number CYCCNT, if the following equations are fulfilled.

$$FR_PSR0[PROTSTATE] = POC:\ normalactive$$

Equation 29

$$FR_MTSACRF[MTE] = 1$$

Equation 30

$$CYCCNT \,\&\, FR_MTSACFR[CYCCNTMSK] = FR_MTSACFR[CYCCNTVAL] \,\&\, FR_MTSACFR[CYCCNTMSK]$$

Equation 31

The MTS is transmitted over channel B in the communication cycle with number CYCCNT, if the following equations are fulfilled.

$$FR_MTSBCRF[MTE] = 1$$

Equation 32

$$CYCCNT \,\&\, FR_MTSBCFR[CYCCNTMSK] = FR_MTSBCFR[CYCCNTVAL] \,\&\, FR_MTSBCFR[CYCCNTMSK]$$

Equation 33

47.8.14 Key Slot Transmission

Key slot assignment, transmission in POC:startup state, and transmission in POC:normal active state are discussed in this section.

47.8.14.1 Key Slot Assignment

A key slot is assigned to the CC if the key_slot_id field in the Protocol Configuration Register 18 (FR_PCR18) is configured with a value greater than 0 and less or equal to number_of_static_slots in Protocol Configuration Register 2 (FR_PCR2), otherwise no key slot is assigned.

47.8.14.2 Key Slot Transmission in POC:startup

If a key slot is assigned and the CC is in the *POC:startup* state, startup null frames will be transmitted as specified by FlexRay Communications System Protocol Specification, Version *2.1 Rev A* .

47.8.14.3 Key Slot Transmission in POC:normal active

If a key slot is assigned and the CC is in *POC:normal active*, a frame of the type as shown in the following table is transmitted. If a transmit message buffer is configured for the key slot and a valid message is available, a message frame is transmitted (see Message Transmission). If no transmit message buffer is configured for the key slot or no valid message is available, a null frame is transmitted (see Null Frame Transmission).

Table 47-53. Key Slot Frame Type

FR_PCR11[key_slot_used_for_sync]	FR_PCR11[key_slot_used_for_startup]	key slot frame type
0	0	normal frame
0	1	normal frame1[1]
1	0	sync frame
1	1	startup frame

1. The frame transmitted has an semantically incorrect header and will be detected as an invalid frame at the receiver.

47.8.15 Sync Frame Filtering

Each received synchronization frame must pass the Sync Frame Acceptance Filter and the Sync Frame Rejection Filter before it is considered for clock synchronization. If the synchronization frame filtering is globally disabled, i.e. the SFFE control bit in the Module Configuration Register (FR_MCR) is cleared, all received synchronization

frames are considered for clock synchronization. If a received synchronization frame did not pass at least one of the two filters, this frame is processed as a normal frame and is not considered for clock synchronization.

47.8.15.1 Sync Frame Acceptance Filtering

The synchronization frame acceptance filter is implemented as a value-mask filter. The value is configured in the Sync Frame ID Acceptance Filter Value Register (FR_SFIDAFVR) and the mask is configured in the Sync Frame ID Acceptance Filter Mask Register (FR_SFIDAFMR). A received synchronization frame with the frame ID FID passes the sync frame acceptance filter, if the following equations evaluates to true.

$$FR_MCR[SFFE] = 0$$

Equation 34

$$FID \,\&\, FR_SFIDAFMR[FMSK] = FR_SFIDAFVR[FVAL] \,\&\, FR_SFIDAFMR[FMSK]$$

Equation 35

Note

Sync frames are transmitted in the static segment only. Thus FID <= 1023.

47.8.15.2 Sync Frame Rejection Filtering

The synchronization frame rejection filter is a comparator. The compare value is defined by the Sync Frame ID Rejection Filter Register (FR_SFIDRFR). A received synchronization frame with the frame ID FID passes the sync frame rejection filter if the following equations evaluates to true.

$$FR_MCR[SFFE] = 0$$

Equation 36

$$FID \neq FR_SFIDRFR[SYNFRID]$$

Equation 37

Note

Sync frames are transmitted in the static segment only. Thus FID <= 1023.

47.8.16 Strobe Signal Support

The CC provides a number of strobe signals for observing internal protocol timing related signals in the protocol engine. The signals are listed and described in the Strobe Signal Mapping table, located in the Register Descriptions section.

47.8.16.1 Strobe Signal Assignment

Each of the strobe signals listed in the Strobe Signal Mapping table, located in the Register Descriptions section, can be assigned to one of the four strobe ports using the Strobe Signal Control Register (FR_STBSCR). To assign multiple strobe signals, the application must write multiple times to the Strobe Signal Control Register (FR_STBSCR) with appropriate settings.

To read out the current settings for a strobe signal with number N, the application must execute the following sequence.

1. Write to FR_STBSCR with WMD = 1 and SEL = N. (updates SEL field only)
2. Read STBCSR.

   The SEL field provides N and the ENB and STBPSEL fields provides the settings for signal N.

47.8.16.2 Strobe Signal Timing

This section provides detailed timing information of the strobe signals with respect to the protocol engine clock.

The strobe signals display internal PE signals. Due to the internal architecture of the PE, some signals are generated several PE clock cycles before the actual action is performed on the FlexRay Bus. These signals are listed in the Strobe Signal Mapping table, located in the Register Descriptions section, with a negative clock offset. An example waveform is given in the following figure.

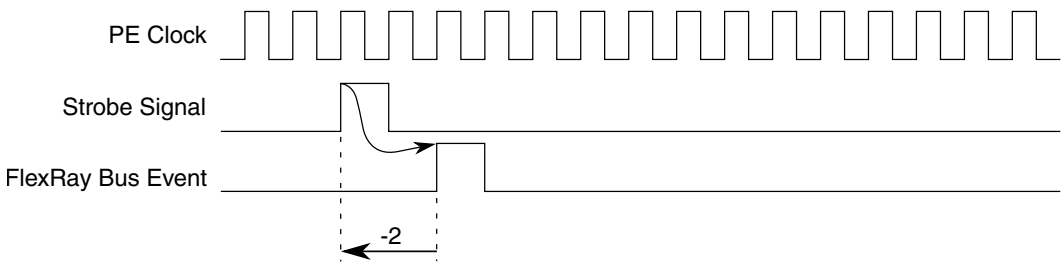


Figure 47-29. Strobe Signal Timing (type = pulse, clk_offset = -2)

Other signals refer to events that occurred on the FlexRay Bus some cycles before the strobe signal is changed. These signals are listed in the Strobe Signal Mapping table, located in the Register Descriptions section, with a positive clock offset. An example waveform is given in the figure below.

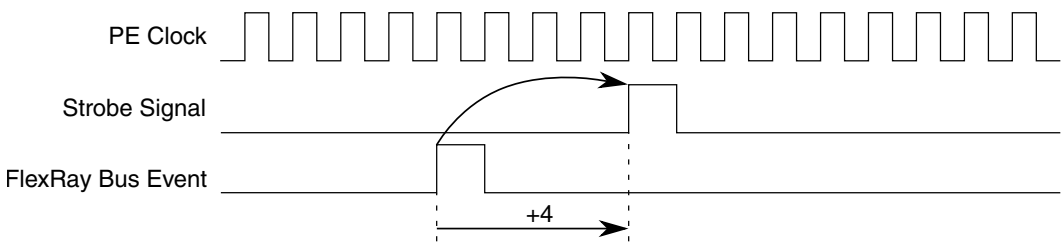


Figure 47-30. Strobe Signal Timing (type = pulse, clk_offset = +4)

47.8.17 Timer Support

The CC provides two timers, which run on the FlexRay time base. Each timer generates a maskable interrupt when it reaches a configured point in time. Timer T1 is an absolute timer. Timer T2 can be configured to be an absolute or a relative timer. Both timers can be configured to be repetitive. In the non-repetitive mode, timer stops if it expires. In repetitive mode, timer is restarted when it expires.

Both timers are active only when the protocol is in *POC:normal active* or *POC:normal passive* state. If the protocol is not in one of these modes, the timers are stopped. The application must restart the timers when the protocol has reached the *POC:normal active* or *POC:normal passive* state.

47.8.17.1 Absolute Timer T1

The absolute timer T1 has the protocol cycle count and the macrotick count as the time base. The timer 1 interrupt flag TI1_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set at the macrotick start event, if the following equations are fulfilled.

$$CYCTR[CTCCNT] \& FR_TI1CYSR[T1_CYC_MSK] = FR_TI1CYSR[T1_CYC_VAL] \& FR_TI1CYSR[T1_CYC_MSK]$$

Equation 38

$$FR_MTCTR[MTCT] = FR_TI1MTOR[T1_MTOFFSET]$$

Equation 39

If the timer 1 interrupt enable bit TI1_IE in the Protocol Interrupt Enable Register 0 (FR_PIER0) is asserted, an interrupt request is generated.

The status bit T1ST is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the T1ST bit is not cleared and the timer is restarted immediately. The T1ST is cleared when the timer is stopped.

The corresponding Interrupt condition (i.e., when the abovementioned equations are fulfilled) also leads to an event out indication on an output port (*fr_evt_tim1*) based on the [TIM1_EE] bit of the Protocol Event Output Enable Register (FR_PEOER).

47.8.17.2 Absolute / Relative Timer T2

The timer T2 can be configured to be an absolute or relative timer by setting the T2_CFG control bit in the Timer Configuration and Control Register (FR_TICCR). The status bit T2ST is set when the timer is triggered, and is cleared when the timer expires and is non-repetitive. If the timer expires but is repetitive, the T2ST bit is not cleared and the timer is restarted immediately. The T2ST is cleared when the timer is stopped.

47.8.17.2.1 Absolute Timer T2

If timer T2 is configured as an absolute timer, it has the same functionality timer T1 but the configuration from Timer 2 Configuration Register 0 (FR_TI2CR0) and Timer 2 Configuration Register 1 (FR_TI2CR1) is used. On expiration of timer T2, the interrupt flag TI2_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set. If the timer 1 interrupt enable bit TI1_IE in the Protocol Interrupt Enable Register 0 (FR_PIER0) is asserted, an interrupt request is generated.

The corresponding Interrupt condition for T2 also leads to an event out indication on an output port (*fr_evt_tim2*) based on the [TIM2_EE] bit of the Protocol Event Output Enable Register (FR_PEOER).

47.8.17.2.2 Relative Timer T2

If the timer T2 is configured as a relative timer, the interrupt flag TI2_IF in the Protocol Interrupt Flag Register 0 (FR_PIFR0) is set, when the programmed amount of macroticks MT[31:0], defined by Timer 2 Configuration Register 0 (FR_TI2CR0) and Timer 2 Configuration Register 1 (FR_TI2CR1), has expired since the trigger or restart of timer 2. The relative timer is implemented as a down counter and expires when it has reached 0. At the macrotick start event, the value of MT[31:0] is checked and then decremented. Thus, if the timer is started with MT[31:0] == 0, it expires at the next macrotick start.

The corresponding Interrupt condition for T2 also leads to an event out indication on an output port (*fr_evt_tim2*) based on the [TIM2_EE] bit of the Protocol Event Output Enable Register (FR_PEOER).

47.8.18 Slot Status Monitoring

The CC provides several means for slot status monitoring. All slot status monitors use the same slot status vector provided by the PE. The PE provides a slot status vector for each static slot, for each dynamic slot, for the symbol window, and for the NIT, on a per channel base. The content of the slot status vector is described in the "Slot Status Content" table below. The PE provides the slot status vector within the first macrotick after the end of the related slot/window/NIT, as shown in the below figure.

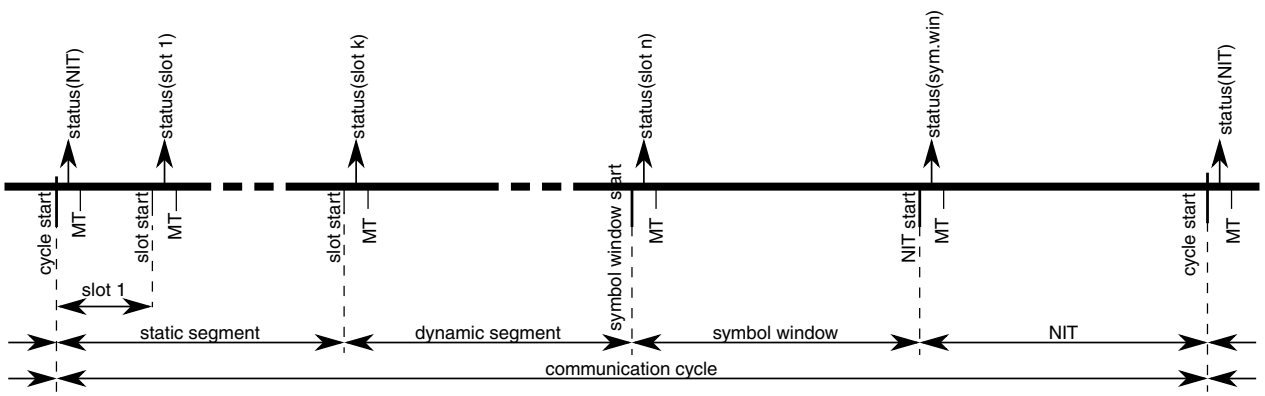


Figure 47-31. Slot Status Vector Update

Note

The slot status for the NIT of cycle n is provided after the start of cycle n+1.

Table 47-54. Slot Status Content

	Status Content
static /dynamic Slot	**slot related status**

Table continues on the next page...

Table 47-54. Slot Status Content (continued)

	Status Content
	vSS!ValidFrame - valid frame received
	vSS!SyntaxError - syntax error occurred while receiving
	vSS!ContentError - content error occurred while receiving
	vSS!BViolation - boundary violation while receiving
	for slots in which the module transmits:
	vSS!TxConflict - reception ongoing while transmission starts
	for slots in which the module does not transmit:
	vSS!TxConflict - reception ongoing while transmission starts
	first valid - channel that has received the first valid frame
	received frame related status
	extracted from
	a)header of valid frame, if vSS!ValidFrame = 1
	b) last received header, if vSS!ValidFrame = 0
	c) set to 0, if nothing was received
	vRF!Header!NFIndicator - Null Frame Indicator (0 for null frame)
	vRF!Header!SuFIndicator - Startup Frame Indicator
	vRF!Header!SyFIndicator - Sync Frame Indicator
Symbol Window	**window related status**
	vSS!ValidFrame - always 0
	vSS!ContentError - content error occurred while receiving
	vSS!SyntaxError - syntax error occurred while receiving
	vSS!BViolation - boundary violation while receiving
	vSS!TxConflict - reception ongoing while transmission starts
	received symbol related status
	vSS!ValidMTS - valid Media Test Access Symbol received
	received frame related status
	see static/dynamic slot
NIT	**NIT related status**
	vSS!ValidFrame - always 0
	vSS!ContentError - content error occurred while receiving
	vSS!SyntaxError - syntax error occurred while receiving
	vSS!BViolation - boundary violation while receiving
	vSS!TxConflict - always 0
	received frame related status
	see static/dynamic slot

47.8.18.1 Channel Status Error Counter Registers

The two channel status error counter registers, Channel A Status Error Counter Register (FR_CASERCR) and Channel B Status Error Counter Register (FR_CBSERCR), incremented by one, if at least one of four slot status error bits, *vSS!SyntaxError, vSS! ContentError, vSS!BViolation*, or *vSS!TxConflict* is set to 1. The status vectors for all slots in the static and dynamic segment, in the symbol window, and in the NIT are taken into account. The counters wrap round after they have reached the maximum value.

47.8.18.2 Protocol Status Registers

The Protocol Status Register 2 (FR_PSR2) provides slot status information about the Network Idle Time NIT and the Symbol Window. The Protocol Status Register 3 (FR_PSR3) provides aggregated slot status information.

47.8.18.3 Slot Status Registers

The eight slot status registers, Slot Status Registers (FR_SSR0–FR_SSR7), can be used to observe the status of static slots, dynamic slots, the symbol window, or the NIT without individual message buffers. These registers provide all slot status related and received frame / symbol related status information, as given in Table 47-54, except of the *first valid* indicator for non-transmission slots.

47.8.18.4 Slot Status Counter Registers

The CC provides four slot status error counter registers, Slot Status Counter Registers (FR_SSCR0–FR_SSCR3). Each of these slot status counter registers is updated with the value of an internal slot status counter at the start of a communication cycle. The internal slot status counter is incremented if its increment condition, defined by the Slot Status Counter Condition Register (FR_SSCCR), matches the status vector provided by the PE. All static slots, the symbol window, and the NIT status are taken into account. *Dynamic* slots are *excluded*. The internal slot status counting and update timing is shown in the figure below.

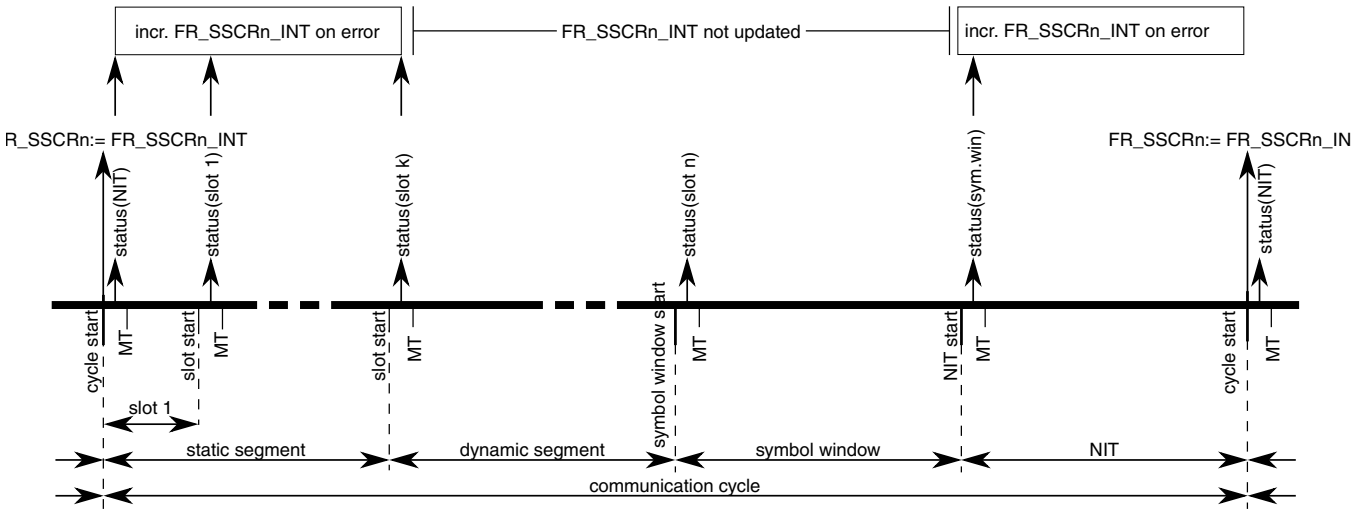


Figure 47-32. Slot Status Counting and FR_SSCRn Update

The PE provides the status of the NIT in the first slot of the next cycle. Due to these facts, the FR_SSCRn register reflects, in cycle n, the status of the NIT of cycle n-2, and the status of all static slots and the symbol window of cycle n-1.

The increment condition for each slot status counter consists of two parts, the frame related condition part and the slot related condition part. The internal slot status counter FR_SSCRn_INT is incremented if at least one of the conditions is fulfilled:

1. frame related condition:

   (FR_SSCCRn[VFR] | FR_SSCCRn[SYF] | FR_SSCCRn[NUF] | FR_SSCCRn[SUF]) // count on frame condition

   = 1;

   and

   ((~FR_SSCCRn[VFR] | vSS!ValidFrame) & // valid frame restriction

   (~FR_SSCCRn[SYF] | vRF!Header!SyFIndicator) & // sync frame indicator restriction

   (~FR_SSCCRn[NUF] | ~vRF!Header!NFIndicator) & // null frame indicator restriction

   (~FR_SSCCRn[SUF] | vRF!Header!SuFIndicator)) // startup frame indicator restriction

   = 1;

Note

The indicator bits SYF, NUF, and SUF are valid only when a valid frame was received. Thus it is required to set the VFR always, whenever count on frame condition is used.

slot related condition:

((FR_SSCCRn[STATUSMASK[3]] & vSS!SyntaxError) | // increment on syntax error

(FR_SSCCRn[STATUSMASK[2]] & vSS!ContentError) | // increment on content error

(FR_SSCCRn[STATUSMASK[1]] & vSS!BViolation) | // increment on boundary violation

(FR_SSCCRn[STATUSMASK[0]] & vSS!TxConflict)) // increment on transmission conflict

= 1;

If the slot status counter is in single cycle mode, i.e. FR_SSCCRn[MCY] = 0, the internal slot status counter FR_SSCRn_INT is reset at each cycle start. If the slot status counter is in the multicycle mode, i.e. FR_SSCCRn[MCY] = 1, the counter is not reset and incremented, until the maximum value is reached.

47.8.18.5 Message Buffer Slot Status Field

Each individual message buffer and each FIFO message buffer provides a slot status field, which provides the information shown in Table 47-54 for the static/dynamic slot. The update conditions for the slot status field depend on the message buffer type. Refer to the Message Buffer Update Sections in Individual Message Buffer Configuration Data.

47.8.19 System Bus Access

This section provides a description of the system bus accesses failures and the related CC behavior. System bus access failures may occur when the CC transfers data to or from the FlexRay memory area.

The system bus access failure types are described in System Bus Access Failure Types.

The behavior of the CC after the occurrence of a system bus access failure is described in System Bus Access Failure Response.

47.8.19.1 System Bus Access Failure Types

This section describes the two types of system bus access failures.

47.8.19.1.1 System Bus Illegal Address Access

A system bus illegal address access is detected when the CC has used an illegal or invalid address to access the FlexRay system memory area. There are three conditions which are treated as a system bus illegal address access:

- The system bus subsystem detects an CC access to an illegal system memory address.
- The CC detects the usage of an data field offset with the value of 0.
- The CC detects a memory error while reading a data field offset from the CHI LRAM memory (see CHI LRAM Error Response after CC Read).

If a system bus illegal address access is detected, the CC sets the ILSA_EF flag in the CHI Error Flag Register (FR_CHIERFR).

47.8.19.1.2 System Bus Access Timeout

A system bus access timout is detected if an access to the FlexRay memory area is not finished in time. The timeout value is derived from the SYMATOR[TIMEOUT] setting (see Configure System Memory Access Time-Out Register (FR_SYMATOR)).

If a system bus access timout is detected, the CC sets the SBCF_EF flag in the CHI Error Flag Register (FR_CHIERFR).

47.8.19.2 System Bus Access Failure Response

This section describes the two types of behavior of the CC after the occurrence of a system bus access failure. The actual behavior is defined by the SBFF bit in the Module Configuration Register (FR_MCR).

47.8.19.2.1 Continue after System Bus Access Failure

If the SBFF bit in the Module Configuration Register (FR_MCR) is 0, the CC will continue its operation after the occurrence of the system bus access failure, but will not generate any system bus accesses until the start of the next communication cycle.Since no

data are read from or written to the FlexRay memory area, no messages are received or transmitted. Consequently, none of the individual message buffers or receive FIFOs will be updated until the next communication cycle starts.

If a frame is under transmission when the system bus failure occurs, a correct frame is generated with the remaining header and frame data are replaced by all zeros. Depending on the point in time this can affect the PPI bit, the Header CRC, the Payload Length in case of an dynamic slot, and the payload data. Starting from the next slot in the current cycle, no frames will be transmitted and received, except for the key slot, where a sync or startup null-frame is transmitted, if the key slot is assigned.

If a frame is received when the system bus failure occurs, the reception is aborted and the related receive message buffer is not updated.

Normal operation is resumed after the start of next communication cycle.

47.8.19.2.2 Freeze after System Bus Access Failure

If the SBFF bit in the Module Configuration Register (FR_MCR) is set to 1, the CC will go into the freeze mode immediately after the occurrence of one of the system bus access failures.

47.8.20 Interrupt Support

The CC provides 172 individual interrupt sources and five combined interrupt sources.

47.8.20.1 Individual Interrupt Sources

Message buffer interrupts are discussed in this section.

47.8.20.1.1 Message Buffer Interrupts

The CC provides 128 message buffer interrupt sources.

Each individual message buffer provides an interrupt flag FR_MBCCSRn[MBIF] and an interrupt enable bit FR_MBCCSRn[MBIE]. The CC sets the interrupt flag when the slot status of the message buffer was updated. If the interrupt enable bit is asserted, an interrupt request is generated.

47.8.20.1.2 FIFO Interrupts

The CC provides 2 FIFO interrupt sources.

Each of the 2 FIFO provides a Receive FIFO Almost Full Interrupt Flag. The CC sets the Receive FIFO Almost Full Interrupt Flags (FR_GIFER[FAFBIF], FR_GIFER[FAFAIF]) in the Global Interrupt Flag and Enable Register (FR_GIFER) if the corresponding Receive FIFO fill level exceeds the defined watermark.

47.8.20.1.3 Wakeup Interrupt

The CC provides one interrupt source related to the wakeup.

The CC sets the Wakeup Interrupt Flag FR_GIFER[WUPIF] when it has received a wakeup symbol on the FlexRay bus. The CC generates an interrupt request if the interrupt enable bit FR_GIFER[WUPIE] is asserted.

47.8.20.1.4 Protocol Interrupts

The CC provides 25 interrupt sources for protocol related events. For details, see Protocol Interrupt Flag Register 0 (FR_PIFR0) and Protocol Interrupt Flag Register 1 (FR_PIFR1). Each interrupt source has its own interrupt enable bit.

47.8.20.1.5 CHI Interrupts

The CC provides 16 interrupt sources for CHI related error events. For details, see CHI Error Flag Register (FR_CHIERFR). There is one common interrupt enable bit FR_GIFER[CHIE] for all CHI error interrupt sources.

47.8.20.2 Combined Interrupt Sources

Each combined interrupt source generates an interrupt request only when at least one of the interrupt sources that is combined generates an interrupt request.

47.8.20.2.1 Receive Message Buffer Interrupt

The Receive Message Buffer Interrupt request is generated when at least one of the individual receive message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[RBIE] is set.

47.8.20.2.2 Transmit Message Buffer Interrupt

The Transmit Message Buffer Interrupt request is generated when at least one of the individual transmit message buffers generates an interrupt request MBXIRQ[n] and the interrupt enable bit FR_GIFER[TBIE] is asserted.

47.8.20.2.3 Protocol Interrupt

The Protocol Interrupt request is generated when at least one of the individual protocol interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[PRIE] is set..

47.8.20.2.4 CHI Interrupt

The CHI Interrupt request is generated when at least one of the individual chi error interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[CHIE] is set.

47.8.20.2.5 Module Interrupt

The Module Interrupt request is generated if at least one of the combined interrupt sources generates an interrupt request and the interrupt enable bit FR_GIFER[MIE] is set.

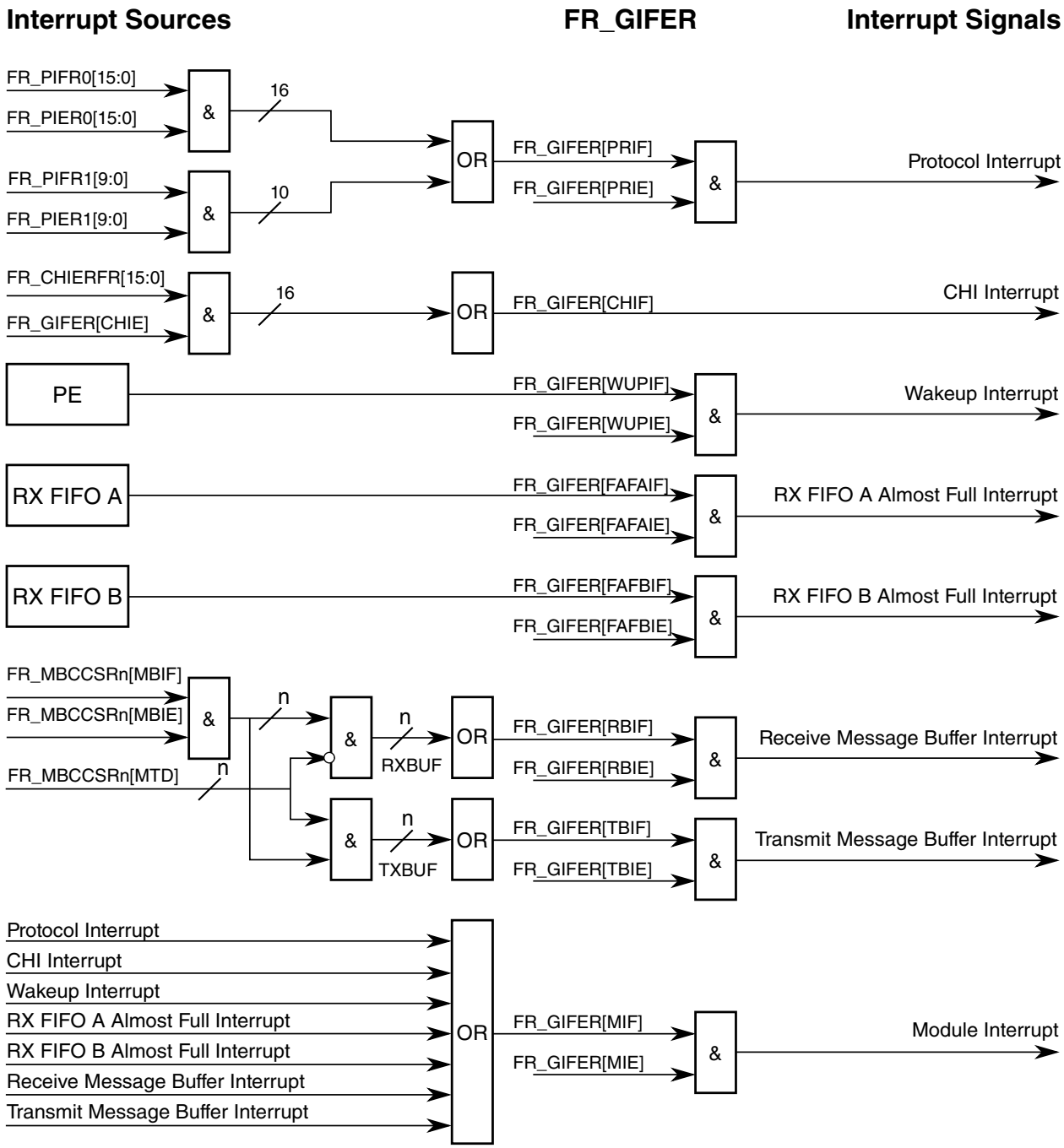


Figure 47-33. Scheme of FR_GIFER interrupt signal generation

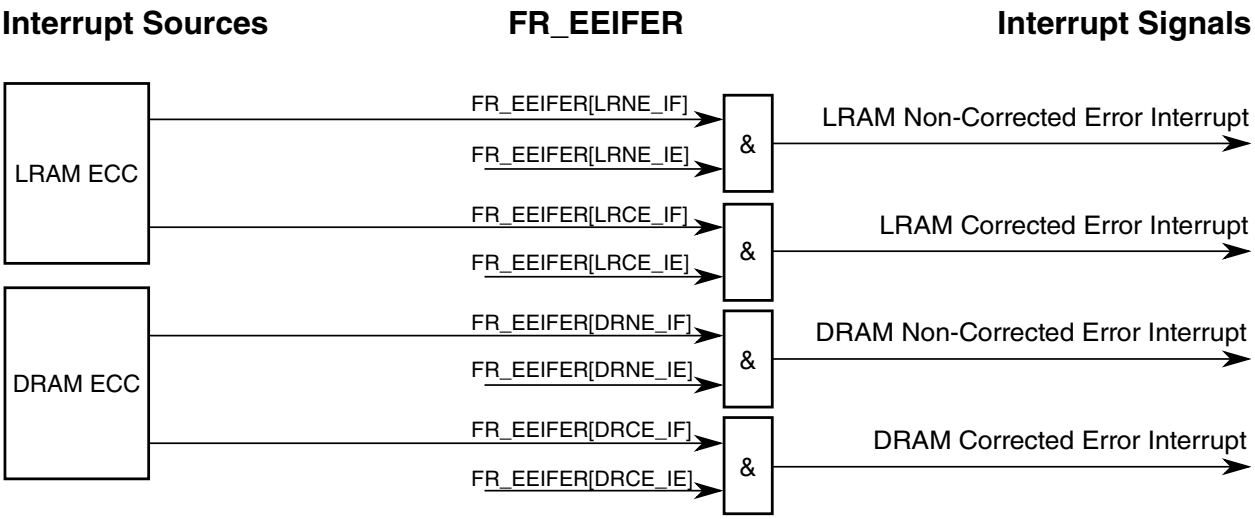


Figure 47-34. Scheme of FR_EEIFER interrupt signal generation

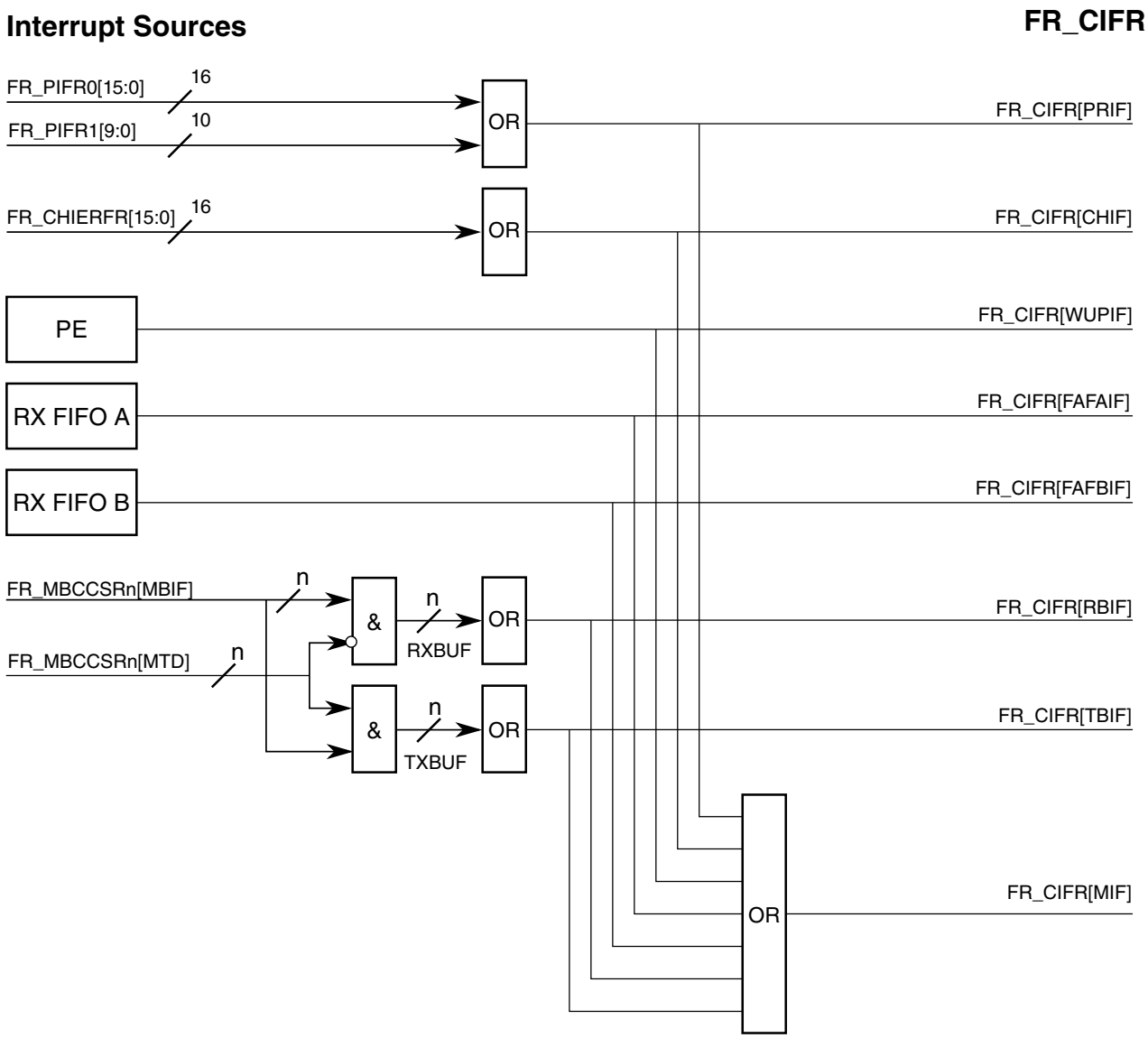


Figure 47-35. Scheme of FR_CIFR flags generation

47.8.21 Lower Bit Rate Support

The CC supports a number of lower bit rates on the FlexRay bus channels. The lower bit rates are implemented by modifying the duration of the microtick *pdMicrotick*, the number of samples per microtick *pSamplesPerMicrotick*, the number of samples per bit *cSamplesPerBit*, and the strobe offset *cStrobeOffset*. The application configures the FlexRay channel bit rate by setting the BITRATE field in the Module Configuration Register (FR_MCR). The protocol values are set internally. The available bit rates, the related BITRATE field configuration settings and related protocol parameter values are shown in the "FlexRay Channel Bit Rate Control" table below.

Table 47-55. FlexRay Channel Bit Rate Control

FlexRay Channel Bit Rate [Mbit/s]	FR_MCR[BITRATE]	*pdMicrotick [ns]*	*gdSampleClockPeriod [ns]*	*pSamplesPerMicrotick*	*cSamplesPerBit*	*cStrobeOffset*
10.0	000	25.0	12.5	2	8	5
8.0	011	25.0	12.5	2	10	6
5.0	001	25.0	25.0	1	8	5
2.5	010	50.0	50.0	1	8	5

Note

The bit rate of 8 Mbit/s is not defined by the *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*.

47.8.22 PE Data Memory (PE DRAM)

The PE Data Memory (PE DRAM) is 128 word, 16-bit wide memory with byte access, which contains the program data of the PE internal CPU. The PE DRAM is divided into two banks, 8-bit each. The memory data [7:0] are assigned to BANK0, the memory data [15:8] are assigned to BANK1.

Table 47-56. PE DRAM Layout

ADDR	BANK1	BANK0
0x00	byte1	byte0
0x01	byte3	byte2
…		
0x7F	byte255	byte254

The FlexRay module provides means to access the PE DRAM from the application. The PE DRAM application access is initiated and controlled via PE DRAM Access Register (FR_PEDRAR) and PE DRAM Access Register (FR_PEDRAR). This functionality is used to check the memory error detection.

47.8.22.1 PE DRAM Read Access

A read access from the PE DRAM can be initiated in any protocol state. The following sequence describes a read access from the PE DRAM address 0x70.

1. FR_PEDRAR:= 0x50E0;

   // INST=0x5; ADDR=070

2. wait until FR_PEDRAR[DAD] == 1;

   // wait for end of PE DRAM access

3. val = FR_PEDRDR[DATA];

   // read PE DRAM data

The read access is handled by the PE internal CPU with the lowest execution priority. This may cause an response delay with a maximum of 1000 PE clock cycle (25us).

47.8.22.2 PE DRAM Write Access

A write access into the PE DRAM can be initiated in any protocol state. The following sequence describes a write access to the PE DRAM address 0x70.

1. FR_PEDRDR:= DATA;

   // write value to be written into data register

2. FR_PEDRAR:= 0x30E0;

   // INST=0x3; ADDR=0x70

3. wait until FR_PEDRAR[DAD] == 1;

   // wait for end of PE DRAM access

4. val = FR_PEDRDR[DATA];

   // read back PE DRAM data

The write access is handled by the PE internal CPU with the lowest execution priority. This may causes an response delay with a maximum of 1000 PE clock cycle (25us).

If the conditions given in PE DRAM Write Access Limitations are fulfilled, the data provided in PE DRAM Access Regsiter (FR_PEDRAR) are written into the PE DRAM, read back in the next clock cycle and stored into the PE DRAM Access Register (FR_PEDRAR). Otherwise, data are not written into the PE DRAM and 0x0000 is stored into the PE DRAM Access Register (FR_PEDRAR).

47.8.22.3 PE DRAM Write Access Limitations

The PE DRAM is used by the protocol engine if the module is not in *POC:default config* state. The only address not used by the protocol engine is 0x70. To prevent the corruption of protocol engine data the following PE DRAM write access limitations apply for application writes.

1. When the module is in *POC:default config* state, all PE DRAM addresses are writable.
2. When the module is not *in POC:default config* state, only PE DRAM address 0x70 is writable.

47.8.23 CHI Lookup-Table Memory (CHI LRAM)

The CHI Lookup-Table Memory (CHI LRAM) is an CHI internal memory which contains the message buffer configuration data and the data field offsets for the physical message buffers. The configuration data for two message buffers or 6 data field offsets are contained in one memory row. The CHI LRAM is divided into 6 memory BANKs.

Table 47-57. CHI LRAM Layout

ADR	BANK5	BANK4	BANK3	BANK2	BANK1	BANK0
0x00	FR_MBIDXR1	FR_MBFIDR1	FR_MBCCFR1	FR_MBIDXR0	FR_MBFIDR0	FR_MBCCFR0
0x01	FR_MBIDXR3	FR_MBFIDR3	FR_MBCCFR3	FR_MBIDXR2	FR_MBFIDR2	FR_MBCCFR2
…						
0x3F	FR_MBIDXR127	FR_MBFIDR127	FR_MBCCFR127	FR_MBIDXR126	FR_MBFIDR126	FR_MBCCFR126
0x40	FR_MBDOR5	FR_MBDOR4	FR_MBDOR3	FR_MBDOR2	FR_MBDOR1	FR_MBDOR0
…						
0x55	FR_MBDOR131	FR_MBDOR130	FR_MBDOR129	FR_MBDOR128	FR_MBDOR127	FR_MBDOR126
0x56	FR_LEETR5	FR_LEETR4	FR_LEETR3	FR_LEETR2	FR_LEETR1	FR_LEETR0

47.8.23.1 CHI LRAM Read and Write Access

The CHI LRAM is accessed by the application via regular register read and write accesses.

47.8.24 Memory Content Fault Detection

The FlexRay module provides integrated memory content error detection for both the CHI LRAM and PE DRAM, and memory content error correction for the PE DRAM. The memory error detection for the CHI LRAM uses an standard Hamming code with a Hamming distance of 3 and detects all single-bit and double-bit errors (SEDDED) . The memory error detection and correction for the PE DRAM uses an enhanced Hamming code with a Hamming distance of 4 and detects and corrects all single-bit errors and detects all double-bit errors (SECDED).

This section describes the reporting of the occurrence of memory content errors, the reaction of the module on the occurrence, and how the application can inject memory errors in order to trigger the report and response behavior.

47.8.24.1 Memory Error Types

A memory error is the distortion of one or more bits read out of the memory. The reading of the values of all zeros and all ones is considered as an special case. The FlexRay module detects and indicates the memory errors as shown in the "Detected Memory Error Types" table below. The entries on the top have higher priority.

Each memory read access reads out *all* banks of the addressed row, and runs error detection on *all* banks, even in the case that the application has triggered a read from only one bank. This may lead to the reporting of an memory error if at least one bank contains a memory error, even if an error free bank has been read.

Table 47-58. Detected Memory Error Types

Memory	Priority	Memory Data	Indication
CHI LRAM	0 (highest)	All Zero's	Non-Corrected Error
PE DRAM			Non-Corrected Error
CHI LRAM		All One's	Non-Corrected Error
PE DRAM			
CHI LRAM	1 (lowest)	One Bit Flipped	Non-Corrected Error
PE DRAM			Corrected Error

Table continues on the next page...

Table 47-58. Detected Memory Error Types (continued)

Memory	Priority	Memory Data	Indication
CHI LRAM		Two Bits Flipped	Non-Corrected Error
PE DRAM			
CHI LRAM		Three or more Bits Flipped	one out of {No error, Non-Corrected Error}, defined by coding given in CHI LRAM Checkbits and CHI LRAM Syndrome
PE DRAM			one out of {No error, Corrected Error, Non-Corrected Error}, defined by coding given in PE DRAM Checkbits and PE DRAM Syndrome

47.8.24.2 Memory Error Reporting

The memory error reporting is enabled only if the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) is set.

For each of the two memories exists two sets of internal registers to store the detection of one corrected and one non-corrected memory error.

If a memory error is detected, the module checks whether the related error interrupt flag in the ECC Error Interrupt Flag and Enable Register (FR_EEIFER) is set.

- *If the error interrupt flag is set, the related internal error reporting register is not updated and the related error overflow flag is set to 1 to indicate a loss of error condition.*
- *If the error interrupt flag is not set, the internal reporting register is updated and the error interrupt flag is set to 1. If two or more memory errors of the same type are detected, the error for the bank with the lower bank number will be reported, and the error overflow flag will be set to 1.*

If a memory error is detected for at least two banks of one memory, the related error overflow flag is set to 1 to indicate a loss of error condition.

In addition to above the Error indications (corrected / non corrected) along with the failing address is sent out from the module for external error logging and corrective actions by the Software.

47.8.24.2.1 PE DRAM Checkbits

The coding of the checkbits reported in ECC Error Report Code Register (FR_EERCR) for PE DRAM memory errors is shown in Table 47-58. This table shows the implemented enhanced Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 47-59. PE DRAM checkbits coding

CODE	CODE				DATA							
	3	2	1	0	7	6	5	4	3	2	1	0
4[1]	X	X	X	X	X	X	X	X	X	X	X	X
3[2]	-	-	-	-	X	X	X	X	-	-	-	-
2	-	-	-	-	X	-	-	-	X	X	X	-
1	-	-	-	-	-	X	X	-	X	X	-	X
0	-	-	-	-	-	X	-	X	X	-	X	X

1. The checkbit CODE[4] is set to 1 if and only if there is a even number of 1's in columns with X.
2. The checkbits CODE[3]… CODE[0] are set to 1 if and only if there is a odd number of 1's in all columns with X.

This coding of the checkbit ensures that neither 0x000 nor 0xFFF are valid code words and written into the memory.

47.8.24.2.2 PE DRAM Syndrome

The coding of the syndrome reported in ECC Error Report Code Register (FR_EERCR) for PE DRAM memory errors is shown in the following table.

Table 47-60. FR_EERCR[CODE] PE DRAM Syndrome Coding

FR_EERCR[CODE]		Description
[4]	[3:0]	
0x1	0x0	No Error (Never appears in error report registers)
0x0	0x0	If data == 0: Non Corrected Error (Dedicated Handling of All Zero Code Word) If data!= 0: Corrected Error (Parity Bit 4)
0x0	0x1	Corrected Error (Parity Bit 0)
0x0	0x2	Corrected Error (Parity Bit 1)
0x0	0x3	Corrected Error (Data Bit 0)
0x0	0x4	Corrected Error (Parity Bit 2)
0x0	0x5	Corrected Error (Data Bit 1)
0x0	0x6	Corrected Error (Data Bit 2)
0x0	0x7	Corrected Error (Data Bit 3)
0x0	0x8	Corrected Error (Parity Bit 3)
0x0	0x9	Corrected Error (Data Bit 4)

Table continues on the next page...

Table 47-60. FR_EERCR[CODE] PE DRAM Syndrome Coding (continued)

FR_EERCR[CODE]		Description
[4]	[3:0]	
0x0	0xA	Corrected Error (Data Bit 5)
0x0	0xB	Corrected Error (Data Bit 6)
0x0	0xC	Corrected Error (Data Bit 7)
0x0	0xD-0xF	Non-Corrected Error
0x1	0x1-0xF	Non-Corrected Error

47.8.24.2.3 CHI LRAM Checkbits

The coding of the checkbits reported in ECC Error Report Code Register (FR_EERCR) for CHI LRAM memory errors is shown in the table below. This table shows the implemented Hamming code. If the error injection was applied to distort the checkbits, then the distorted checkbits are reported.

Table 47-61. CHI LRAM checkbits coding

CODE[1]	DATA															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	X	X	X	X	X	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	-	-	X	X	X	X	X	X	X	-	-	-	-
2	X	X	-	-	-	X	X	X	X	-	-	-	X	X	X	-
1	-	-	X	X	-	X	X	-	-	X	X	-	X	X	-	X
0	X	-	X	-	X	X	-	X	-	X	-	X	X	-	X	X

1. The checkbit CODE[n] is set to 1 if and only if there is a odd number of 1's in all columns with X.

47.8.24.2.4 CHI LRAM Syndrome

The coding of the syndrome reported in ECC Error Report Code Register (FR_EERCR) for CHI LRAM memory errors is shown in the next table.

Table 47-62. FR_EERCR[CODE] CHI LRAM Syndrome Coding

FR_EERCR[CODE]	Description
0x00	No Error (Never appears in error report registers)
0x01-0x1F	Non Corrected Error

47.8.24.3 Memory Error Response

The memory error response is enabled only when the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) is set.

In case of the detection of a *corrected* memory error, the FlexRay module continues its normal operation using the corrected data word. This section describes the behavior of the FlexRay module after the detection of a *non-corrected* memory error.

47.8.24.3.1 CHI LRAM Error Response after CC Read

When the CC is out of the *POC:default config* state, it reads the configuration data and the data field offsets of all utilized message buffers in every slot and in the NIT. If a non-corrected memory error is detected during this module read access the error response of the module depends from LRAM location where the error occurred.

- *If the LRAM address belongs to physical message buffer configuration data the FlexRay module will consider the affected message buffer as disabled for the current search and will exclude this buffer from the search. The configuration of the affected message buffer is not changed.*

  If the affected message buffer is a tx message buffer, no frame will be transmitted from this message buffer in the next slot. If the affected message buffer is a rx message buffer, no frame will be received to this message buffer in the next slot.

- *If the LRAM address belongs to the data field offset area and the related physical message buffer is used for Rx or Tx the first access to the system memory caused by payload read or write yields to the assertion of the FR_CHIERFR[ILSA_EF]. No memory access occurs w.r.t. payload access is performed for the complete frame.*

47.8.24.3.2 CHI LRAM Error Response after Application Read

The application can read the content of the CHI LRAM via reading the FR_MBCCFRn, FR_MBFIDRn, FR_MBIDXRn, FR_MBDORn, and FR_LEETRn registers. If a non-corrected memory error is detected during this kind of read access, the module indicates the detected memory error, delivers the non-corrected data read and continues its normal operation.

47.8.24.3.3 PE DRAM Error Response after CC Read

If the CC detects an non-corrected memory error during internal read of program data which is contained in PE DRAM, this is considered as an fatal protocol error and the module enters the protocol freeze state immediately.

47.8.24.3.4 PE DRAM Error Response after Application Read in *POC:default config* state

If the CC detects an non-corrected memory error during an application triggered read from any PE DRAM address and the protocol is in the *POC:default config* state, this is considered as an fatal protocol error and the module enters the protocol freeze state. This behavior allows for checking the freeze functionality in case of the detection of non-corrected errors.

47.8.24.3.5 PE DRAM Error Response after Application Read out of *POC:default config*

If the CC detects an non-corrected memory error during an application triggered read from any PE DRAM address, and the protocol is not in the *POC:default config* state, this error is not considered as an fatal error and the protocol state is not changed. This prevents any interference of the running protocol by PE DRAM error injection reads.

47.8.25 Memory Fault Injection

The error injection functionality is used by the application to inject data errors into the memories to trigger and check the memory error detection functionality.

The error injection is enabled only if the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) and the error injection enable control bit EIE in the ECC Error Report and Injection Control Register (FR_EERICR) are set.

The error injection mode is configured by the EIM configuration bit in the ECC Error Report and Injection Control Register (FR_EERICR).

When the error injection is enabled, each write access to the configured memory location will be distorted.

The injector has the same behavior for FlexRay module memory writes and application memory writes.

47.8.25.1 CHI LRAM Error Injection

The following sequence describes an memory error injection sequence for the CHI LRAM memory. This sequence consists of the setup of the error injector followed by an application triggered write access to provoke an distortion of the memory content. The content of the CHI LRAM is described in Table 47-57.

When the CC is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config* see Memory Fault Injection out of POC:default config.

Injector Setup:

1. FR_MCR[ECCE]:= 1;

   // enable ECC functionality

2. FR_EERICR[EIE]:=I_MODE;

   // configure error injection mode

3. FR_EEIAR[MID]:= 1;

   // select CHI LRAM for error injection

4. FR_EEIAR[BANK]:= I_BANK;

   // select bank for error injection; I_BANK = {0,1,2,3,4,5}

5. FR_EEIAR[ADDR]:= I_ADDR;

   // select address for error injection; I_ADDR <= 0x 56

6. FR_EEIDR[DATA]:= D_DIST;

   // define data distortion pattern

7. FR_EEICR[CODE]:= C_DIST;

   // define checkbit distortion pattern

8. FR_EERICR[EIE]:=1;

   // enable error injection

Application Write Access:

If (I_BANK==0) -> FR_MBCCFR(2n) / FR_MBDOR(6k) / FR_LEETR0 := DATA;

If (I_BANK==1) -> FR_MBFIDR(2n) / FR_MBDOR(6k+1) / FR_LEETR1 := DATA;

If (I_BANK==2) -> FR_MBIDXR(2n) / FR_MBDOR(6k+2) / FR_LEETR2 := DATA;

If (I_BANK==3) -> FR_MBCCFR(2n+1) / FR_MBDOR(6k+3) / FR_LEETR3 := DATA;

If (I_BANK==4) -> FR_MBFIDR(2n+1) / FR_MBDOR(6k+4) / FR_LEETR4 := DATA;

If (I_BANK==5) -> FR_MBIDXR(2n+1) / FR_MBDOR(6k+5) / FR_LEETR5 := DATA;

// write DATA to the defined injection bank and injection address (see Table 47-57).

47.8.25.2 PE DRAM Error Injection

The following sequence describes an memory error injection sequence for the PE DRAM memory. This sequence consists of the setup of the error injector followed by an application triggered write access to provoke an distortion of the memory content.

When the FlexRay module is in *POC:default config*, there are no limitations for the error injection and no impacts of error injection to the application. For error injection out of *POC:default config* see PE DRAM Error Injection out of POC:default config.

Injector Setup:

1. FR_MCR[ECCE]:= 1;

   // enable ECC functionality

2. FR_EERICR[EIE]:=I_MODE;

   // configure error injection mode

3. FR_EEIAR[MID]:= 0;

   // select PE DRAM for error injection

4. FR_EEIAR[BANK]:= I_BANK;

   // define bank for error injection; I_BANK = {0,1}

5. FR_EEIAR[ADDR]:= I_ADDR;

   // define address for error injection; I_ADDR <= 0x7F

6. FR_EEIDR[DATA]:= D_DIST;

   // define data distortion pattern

7. FR_EEICR[CODE]:= C_DIST;

   // define checkbit distortion pattern

8. FR_EERICR[EIE]:=1;

   // enable error injection

Application Write Access (e.g. I_ADDR=0x70):

1. FR_PEDRAR:= 0x30E0;

   // INST=0x3; ADDR=0x70

2. wait until FR_PEDRAR[DAD] == 1;

   // wait for end of PE DRAM access

3. val = FR_PEDRDR[DATA]; |

   // get read back PE DRAM data

Note

The write access to the PE DRAM triggers an subsequent read access from PE DRAM in the next cycle, which triggers the detection of the distorted data.

47.8.26 StopWatch function

FlexRay supports a StopWatch function to facilitate OS synchronization to the FlexRay network. This provides a capability to capture the time difference in system clocks between the operating system timer tick and the FlexRay time base. This allows the software synchronization handler to adjust the OS clock to sync with the FR macrotick and are fulfilled.

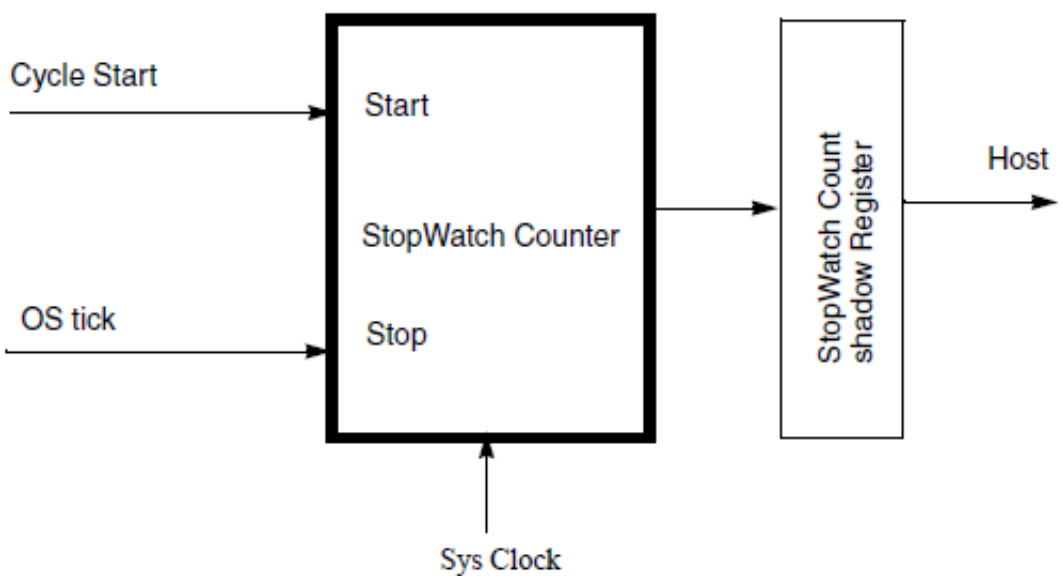


Figure 47-36. StopWatch counter logic

As shown below, there is a 32 bit stopwatch counter that resets and starts with each cycle start. This counter runs on the system clock and increments every cycle until an OS tick event is received, after which the counter halts and the count value is transferred to a shadow register. The shadow register (Stop Watch Count Register (FR_STPWR)) holds the counter value till the host reads all the 32 bits, after which it automatically gets cleared. The StopWatch logic is controlled through a control bit STPW_EN in the FR_PEOER (Protocol Event Output Enable and StopWatch Control Register (FR_PEOER). The counter runs only when this bit is set. The following figure shows the actual timing of the stopwatch counter logic operation.

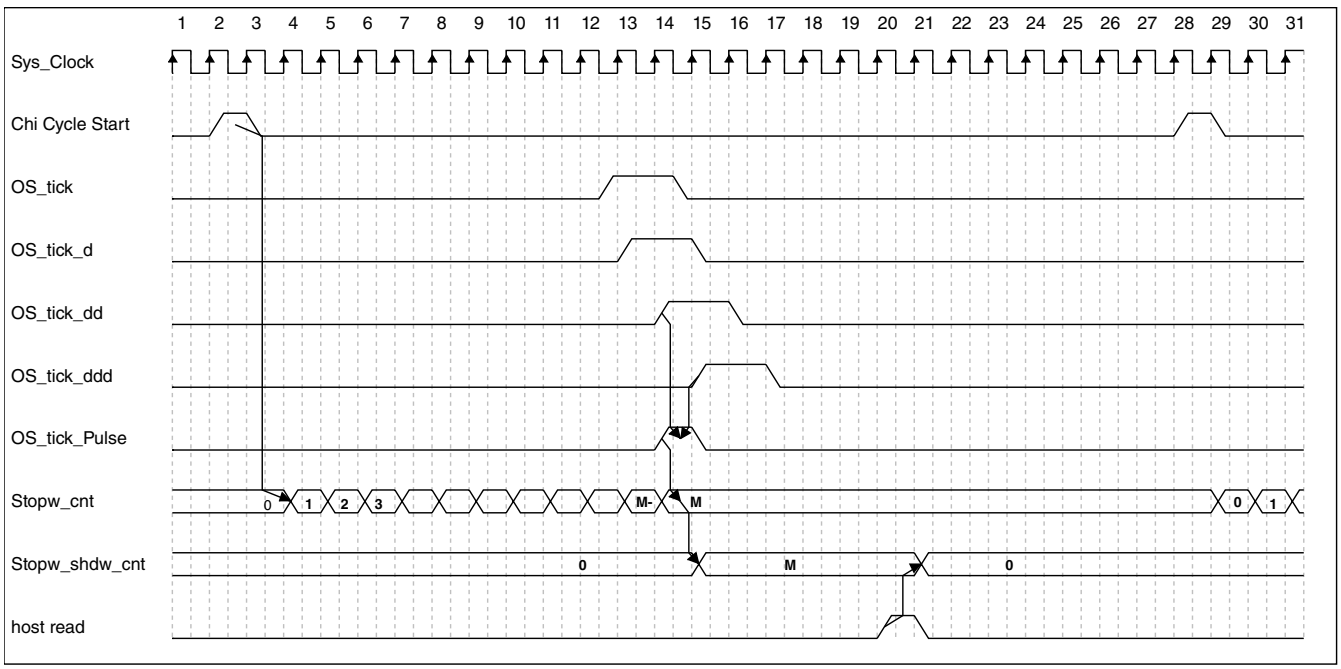


Figure 47-37. StopWatch function timing diagram

NOTE

'Host read' implies that all 32 bits have been read out completely. Any partial read (i.e for 8/16 bits access) will not clear the StopWatch count register.

NOTE

If cycle start and OS tick occurs simultaneously, timer will remain at 0. But OS_tick has a 2 stage synchronizer and cycle_start has a single stage synchronizer into chi clock domain. So simultaneously here means OS_tick_pulse and chi_cycle_start occurring at the same cycle.

47.9 Application Information

Module operation is discussed in this section.

47.9.1 Module Configuration

This section describes essential parts of the module configuration.

47.9.1.1 Configure System Memory Access Time-Out Register (FR_SYMATOR)

To ensure reliable operation of the CC, the application must ensure that the TIMEOUT value in System Memory Access Time-Out Register (FR_SYMATOR) and the CHI clock frequency f_{CHI} in MHz fulfill this equation[1] :

$$0 \le \mathrm{SYMATOR}[\mathrm{TIMEOUT}] \le \lfloor 0.45 \cdot f_{\mathrm{CHI}} - 8 \rfloor$$

Equation 40

For a given SYMATOR[TIMEOUT] value, f_{CHI} can be increased without causing unreliable operation of the CC. The same holds for reducing the SYMATOR[TIMEOUT] value for a given f_{CHI}.

Some examples for maximum values of the SYMATOR[TIMEOUT] for a minimum CHI frequency are given in the table below.

Table 47-63. Maximum SYMATOR[TIMEOUT] examples

f_{CHI}	SYMATOR[TIMEOUT]	f_{CHI}	SYMATOR[TIMEOUT]
>= 18 MHz	0	>= 100 MHz	<= 37
>= 23 MHz	<= 2	>= 120 MHz	<= 46
>= 27 MHz	<= 4	>= 140 MHz	<= 55
>= 32 MHz	<= 6	>= 160 MHz	<= 64
>= 60 MHz	<= 19	>= 180 MHz	<= 73
>= 80 MHz	<= 28	>= 200 MHz	<= 82

1. See Controller Host Interface Clocking for all constraints of minimum CHI clock frequency.

47.9.1.1.1 System Bus Wait State Constraints

The SYMATOR[TIMEOUT] value corresponds directly to a certain acceptable number of wait states on the system bus.

For single channel configurations and if the sync frame table generation functionality is *not* used (FR_SFTCCSR[SDVEN,SIDEN] = 0) no timeout will be detected if less than 2 × SYMATOR[TIMEOUT]+1 wait states will be seen on the system bus for each system bus access.

For dual channel configurations, or if the sync frame table generation functionality is used, no timeout will be detected if less than SYMATOR[TIMEOUT]-1 wait states will be seen on the system bus for each system bus access.

47.9.1.2 Configure Data Field Offsets

The data field offsets are located in the Message Buffer Data Field Offset Registers (FR_MBDORn) and Receive FIFO Start Data Offset Register (FR_RFSDOR). The application has to configure the data field offset values for all message buffers which are used.

When the module is enabled, the initilization value of the FR_MBDORn[MBDO] and FR_RFSDOR[SDO] is 0. This value is considered to be illegal (see System Bus Illegal Address Access).

47.9.2 Initialization Sequence

This section describes the required steps to initialize the CC. The first subsection describes the steps required after a module reset, the second section describes the steps required after preceding shutdown of the CC.

47.9.2.1 Module Initialization

This section describes the module related initialization steps after a system reset.

1. Configure CC.
    a. configure the control bits in the Module Configuration Register (FR_MCR)
    b. configure the system memory base address in System Memory Base Address Register (FR_SYMBADR)
2. Enable the CC.

a. write 1 to the module enable bit MEN in the Module Configuration Register (FR_MCR)

The CC now enters the Normal Mode. The application can commence with the protocol initialization described in Protocol Initialization.

47.9.2.2 Protocol Initialization

This section describes the protocol related initialization steps.

1. Configure the Protocol Engine.
   a. issue CONFIG command via Protocol Operation Control Register (FR_POCR)
   b. wait for *POC:config* in Protocol Status Register 0 (FR_PSR0)
   c. configure the FR_PCR0,…, FR_PCR30 registers to set all protocol parameters
2. Configure the Message Buffers and FIFOs.
   a. set the number of message buffers used and the message buffer segmentation in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR)
   b. define the message buffer data size in the Message Buffer Data Size Register (FR_MBDSR)
   c. configure each message buffer by setting the configuration values in the Message Buffer Configuration, Control, Status Registers (FR_MBCCSRn), Message Buffer Cycle Counter Filter Registers (FR_MBCCFRn), Message Buffer Frame ID Registers (FR_MBFIDRn), Message Buffer Index Registers (FR_MBIDXRn)
   d. configure the FIFOs
   e. issue CONFIG_COMPLETE command via Protocol Operation Control Register (FR_POCR)
   f. wait for *POC:ready* in Protocol Status Register 0 (FR_PSR0)

After this sequence, the CC is configured as a FlexRay node and is ready to integrate into the FlexRay cluster.

47.9.2.3 CHI LRAM Initialization

The initialization of the CHI LRAM is performed by the CC when it leaves the Disabled Mode. The unitization runs for 87 CHI clock cycles. All fields in the FR_MBCCSRn, FR_MBCCFRn, FR_MBFIDRn, FR_MBDORn, and LEETRn registers are initialized to 0. All application read or write accesses to these registers are delayed until the initialization is finished.

47.9.2.4 PE DRAM Initialization

The PE DRAM initialization is performed by the CC in the *POC:default config* state. This initialization runs for 4.8 µs, and will delay the state transition from *POC:default config* into *POC:config*.

47.9.3 Memory Fault Injection out of POC:default config

This section provides information for application driven memory fault injection out if *POC:default config*. The CC provides means to inject memory faults from the application without any impacts to the internal protocol operation of the CC.

47.9.3.1 CHI LRAM Error Injection out of POC:default config

The CC will never perform any internal read access from the LRAM ECC Error Test Registers (FR_LEETRn). Any memory errors injected into these CHI LRAM locations will never be detected by internal access, independent from the protocol state.

The application should use these registers and related CHI LRAM location to inject memory errors into the CHI LRAM. The injection sequence is described in CHI LRAM Error Injection.

47.9.3.2 PE DRAM Error Injection out of POC:default config

The CC will never perform any internal read access from the PE DRAM address 0x70. This is the only one PE DRAM address writable by the application out of the *POC:default config* state.

The application should use these PE DRAM location to inject memory errors into the PE DRAM. The injection sequence is described in PE DRAM Error Injection.

47.9.4 Shut Down Sequence

This section describes a secure shut down sequence to stop the CC gracefully. The main targets of this sequence are

- *finish all ongoing reception and transmission*
- *do not corrupt FlexRay bus and do not disturb ongoing FlexRay bus communication*

For a graceful shutdown the application shall perform the following tasks:

1. Disable all enabled message buffers.
   a. repeatedly write 1 to FR_MBCCSRn[EDT] until FR_MBCCSRn[EDS] == 0.
2. Stop Protocol Engine.
   a. issue HALT command via Protocol Operation Control Register (FR_POCR)
   b. wait for *POC:halt* in Protocol Status Register 0 (FR_PSR0)

47.9.5 Number of Usable Message Buffers

This section describes the required minimum CHI clock frequency for a specified number of utilized message buffers configured in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR), a configured mini-slot length *gdMinislot*, and a configured nominal macrotick length *gdMacrotick*[1].

Additional constraints for the minimum CHI clock frequency are given in Controller Host Interface Clocking.

The CC uses a sequential search algorithm to determine the individual message buffer assigned or subscribed to the next slot. This search is started at the start of slot and must be finished before the start of the next slot.

The shortest FlexRay slot is a corrected empty dynamic slot. An corrected empty dynamic slot is a mini-slot and consists of gdMinislot corrected macroticks with a duration of *gdMacrotick*. The minimum duration of an corrected macrotick is $gdMacrotick_{min}$ = 39 µT. This results in a minimum length of an correct slot

$$\Delta_{slotmin} = 39 \bullet pdMicrotick \bullet gdMinislot$$

Equation 41

1. See Controller Host Interface Clocking for all constraints of minimum CHI clock frequency.

The message buffer search engine runs on the CHI clock and evaluates one individual message buffer per CHI clock cycle. For internal status update operations and to account for clock domain crossing jitter, an additional amount of 27 CHI clock cycles is required to ensure correct search engine operation.

For a given number of utilized message buffers FR_MBSSUTR[LAST_MB_UTIL] + 1 and for a given CHI clock frequency f_{chi}, this results in a search duration of

$$\Delta_{search} = \frac{1}{f_{chi}} \bullet (FR_MBSSUTR[LAST_MB_UTIL] + 27)$$

Equation 42

The message buffer search must be finished within one slot which requires fulfillment of this equation::

$$\Delta_{search} \leq \Delta_{slotmin}$$

Equation 43

This results in the formula given below which determines the required minimum CHI frequency for a given number of message buffers that are utilized.

$$f_{chi} \geq \frac{(FR_MBSSUTR[LAST_MB_UTIL]+27)}{39 \bullet pdMicrotick \bullet gdMinislot}$$

Equation 44

The required minimum CHI Clock frequency for a selected set of relevant protocol parameters and for the LAST_MB_UTIL field in the Message Buffer Segment Size and Utilization Register (FR_MBSSUTR) set to 127 is given in the following table.

Table 47-64. Minimum fchi [MHz] examples (128 message buffers used)

pdMicrotick [ns]	*gdMinislot*					
	2	**3**	**4**	**5**	**6**	**7**
25.0	79.5	53	39.8	31.8	26.5	22.8
50.0	39.8	26.5	19.9	15.9	13.3	11.4

Note

If the minimum CHI frequency is not met the CHIERFR[MBS_EF] flag is set. Refer to the CHI Error Flag Register (FR_CHIERFR) for details.

47.9.6 Protocol Control Command Execution

This section considers the issues of the protocol control command execution.

The application issues any of the protocol control commands listed in the POCCMD field of the FR_POCR Field Descriptions table, located in the Register Descriptions section, by writing the command to the POCCMD field of the Protocol Operation Control Register (FR_POCR). As a result the CC sets the BSY bit while the command is transferred to the PE. When the PE has accepted the command, the BSY flag is cleared. All commands are accepted by the PE.

The PE maintains a protocol command vector. For each command that was accepted by the PE, the PE sets the corresponding command bit in the protocol command vector. If a command is issued while the corresponding command bit is set, the command is not queued and is lost.

If the command execution block of the PE is idle, it selects the next accepted protocol command with the highest priority from the current protocol command vector according to the protocol control command priorities given in the next table. If the current protocol state does not allow the execution of this protocol command (see POC state changes in *FlexRay Communications System Protocol Specification, Version 2.1 Rev A*) the CC asserts the illegal protocol command interrupt flag IPC_IF in the Protocol Interrupt Flag Register 1 (FR_PIFR1). The protocol command is not executed in this case.

Some protocol commands may be interrupted by other commands or the detection of a fatal protocol error as indicated by the table below. If the application issues the FREEZE or READY command, or if the PE detects a fatal protocol error, some commands already stored in the command vector will be removed from this vector.

Table 47-65. Protocol Control Command Priorities

Protocol Command	Priority	Interrupted By	Cleared and Terminated By
FREEZE	(highest) 1	none	
READY	2		
CONFIG_COMPLETE	3		
ALL_SLOTS	4	FREEZE, READY, CONFIG_COMPLET, fatal protocol error	FREEZE, READY, CONFIG_COMPLETE,fatal protocol error
ALLOW_COLDSTART	5		
RUN	6		FREEZE, fatal protocol error
WAKEUP	7		FREEZE, fatal protocol error
DEFAULT_CONFIG	8		FREEZE, fatal protocol error
CONFIG	9		
HALT	(lowest) 10		FREEZE, READY, CONFIG_COMPLETE,fatal protocol error

47.9.7 Message Buffer Search On Simple Message Buffer Configuration

This sections describes the message buffer search behavior for a simplified message buffer configuration. The FIFO behavior is not considered in this section.

47.9.7.1 Simple Message Buffer Configuration

A simple message buffer configuration is a configuration that has at most one transmit message buffer and at most one receive message buffer assigned to a slot *S*. The simple configuration used in this section utilizes two message buffers, one single buffered transmit message buffer and one receive message buffer.

The transmit message buffer has the message buffer number *t* and has following configuration

Table 47-66. Transmit Buffer Configuration

Register	Field	Value	Description
FR_MBCCSR*t*	MTD	1	transmit buffer
FR_MBCCFR*t*	MTM	0	event transition mode
	CHA	1	assigned to channel A
	CHB	0	not assigned to channel B
	CCFE	1	cycle counter filter enabled
	CCFMSK	000011	cycle set = {4n} = {0,4,8,12,…}
	CCFVAL	000000	
FR_MBFIDR*t*	FID	*S*	assigned to slot S

The availability of data in the transmit buffer is indicated by the commit bit FR_MBCCSRt[CMT] and the lock bit FR_MBCCSRt[LCKS].

The receive message buffer has the message buffer number r and has following configuration

Table 47-67. Receive Buffer Configuration

Register	Field	Value	Description
FR_MBCCSR*r*	MTD	0	receive buffer
FR_MBCCFR*r*	MTM	-	n/a
	CHA	1	assigned to channel A
	CHB	0	not assigned to channel B
	CCFE	1	cycle counter filter enabled
	CCFMSK	000001	cycle set = {2n} = {0,2,4,6,…}

Table continues on the next page...

Table 47-67. Receive Buffer Configuration (continued)

Register	Field	Value	Description
	CCFVAL	000000	
FR_MBFIDR*r*	FID	*S*	subscribed slot

Furthermore the assumption is that both message buffers are enabled (FR_MBCCSRt[EDS] = 1 and FR_MBCCSRr[EDS] = 1)

Note

The cycle set {4n+2} = {2,6,10,…} is assigned to the receive buffer only.

The cycle set {4n} = {0,4,8,12,…} is assigned to both buffers.

47.9.7.2 Behavior in static segment

In this case, both message buffers are assigned to a slot *S* in the *static* segment.

The configuration of a transmit buffer for a static slot *S* assigns this slot to the node as a transmit slot. The FlexRay protocol requires:

- When a slot occurs, if the slot is assigned to a node on a channel that node must transmit either a normal frame or a null frame on that channel. Specifically, a null frame will be sent if there is no data ready, or if there is no match on a transmit filter (cycle counter filtering, for example).

Regardless of the availability of data and the cycle counter filter, the node will transmit a frame in the static slot *S*. In any case, the result of the message buffer search will be the transmit message buffer *t*. The receive message buffer *r* will not be found, no reception is possible.

47.9.7.3 Behavior in dynamic segment

In this case, both message buffers are assigned to a slot *S* in the *dynamic* segment. The FlexRay protocol requires:

- When a slot occurs, if a slot is assigned to a node on a channel that node only transmits a frame on that channel if there is data ready and there is a match on relevant transmit filters (no null frames are sent).

The transmission of a frame in the dynamic segment is determined by the availability of data and the match of the cycle counter filter of the transmit message buffer.

47.9.7.3.1 Transmit Data Not Available

If transmit data are *not available*, i.e. the transmit buffer is not committed FR_MBCCSR*t*[CMT]=0 and/or locked FR_MBCCSR*t*[LCKS]=1,

1. for the cycles in the set {4n}, which is assigned to both buffers, the receive buffer will be found and the node can receive data, and
2. for the cycles in the set {4n+2}, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive cycles are shown in the following figure.

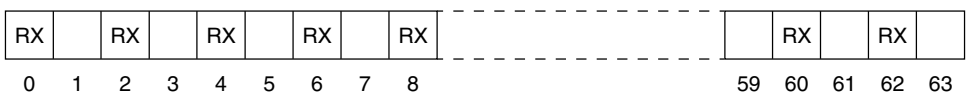


Figure 47-38. Transmit Data Not Available

47.9.7.3.2 Transmit Data Available

If transmit data are *available*, i.e. the transmit buffer is committed FR_MBCCSR*t*[CMT]=1 and not locked FR_MBCCSR*t*[LCKS]=0,

1. for the cycles in the set {4n}, which is assigned to both buffers, the transmit buffer will be found and the node transmits data.
2. for the cycles in the set {4n+2}, which is assigned to the receive buffer only, the receive buffer will be found and the node can receive data.

The receive and transmit cycles are shown in the following figure.

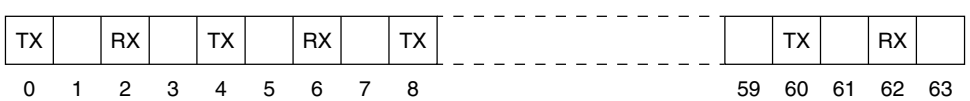


Figure 47-39. Transmit Data Available

Chapter 48
Synchronous Audio Interface (SAI)

48.1 Chip-specific SAI information

48.1.1 SAI configuration

The device instantiates three Synchronous Audio Interface (SAI) modules. The first Synchronous Audio Interface (SAI0) module supports up to 4 channels either with parallel data bits or using the AC97 mode with a single data bit and time division multiplexing (TDM).

In addition two more Synchronous Audio Interfaces are instantiated (SAI1 and SAI2) allowing one channel input or output each. All SAI instances can operate in asynchronous or synchronous mode. In synchronous mode SAI2 can control either SAI1 and SAI0 or only SAI1.

In total, the device can support up to 6 (stereo) audio channels (up to 6 data pins).

Table 48-1. SAI_0 configuration

Feature	Value
Channel supported	4
FIFO size	8
FRAME length	Maximum of 16 words per frame
Frame Synchronization	I2S, AC97, and CODEC/DSP interfaces
Multiple SAI Synchronous Mode	Supported but only SAI_2 can be the master for the multiple SAI synchronous modes.

Table 48-2. SAI_1 and SAI_2 configuration

Feature	Value
Channel supported	1
FIFO size	8

Table continues on the next page...

Table 48-2. SAI_1 and SAI_2 configuration (continued)

Feature	Value
FRAME length	Maximum of 16 words per frame
Frame Synchronization	I2S and CODEC/DSP interfaces
Multiple SAI Synchronous Mode	Supported but only SAI_2 can be the master for the multiple SAI synchronous modes.

48.1.1.1 Timer interface

The timer interface is used to create closed loop control with the Fractional Clock Dividers (FCD)s.

48.1.2 SAI's stop-stop ack protocol

If stop mode request is asserted :

- Transmitter and receiver will complete the existing frame and then disable internal bit clock and then acknowledges the stop request.
- If transmitter and receiver are disabled (TE=0, RE=0) then they will acknowledges the stop request immediately.

48.2 Introduction

The I^2S (or I2S) module provides a synchronous audio interface (SAI) that supports full-duplex serial interfaces with frame synchronization such as I^2S, AC97, TDM, and codec/DSP interfaces.

48.2.1 Features

Note that some of the features are not supported across all SAI instances; see the chip-specific information in the first section of this chapter.

NOTE

About data lines and audio channels: Typically there are one or more data lines for TX and RX sides of the SAI peripheral, depending on the device's design. Each SAI data line may support 1 - 32 audio channels (or audio words).

- Transmitter with independent bit clock and frame sync supporting 4 data lines

- Receiver with independent bit clock and frame sync supporting 4 data lines
- Maximum Frame Size of 16 words
- Word size of between 8-bits and 32-bits
- Word size configured separately for first word and remaining words in frame
- Asynchronous 8 × 32-bit FIFO for each transmit and receive channel
- Supports graceful restart after FIFO error
- Supports automatic restart after FIFO error without software intervention
- Supports packing of 8-bit and 16-bit data into each 32-bit FIFO word
- Supports combining multiple data channel FIFOs into single data channel FIFO

48.2.2 Block diagram

The following block diagram also shows the module clocks.

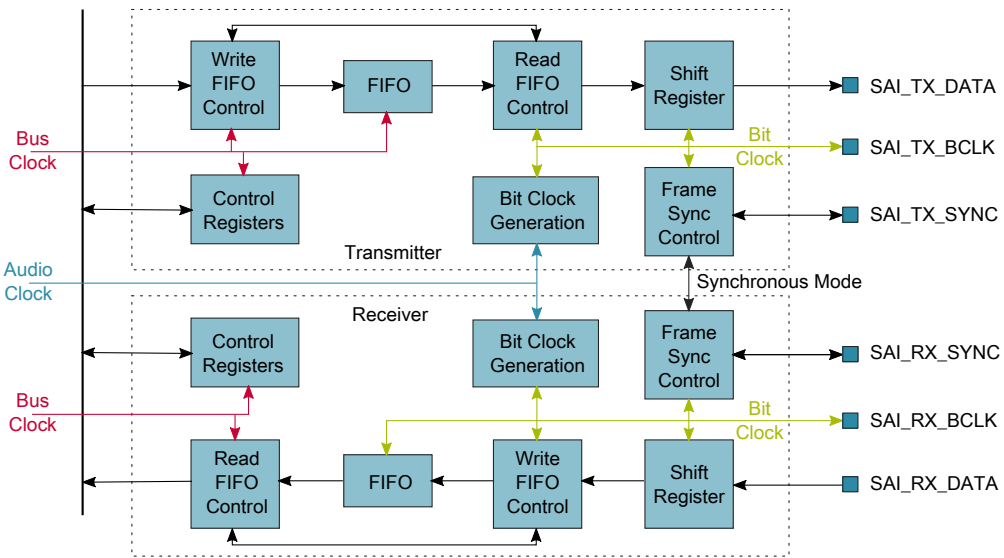


Figure 48-1. I²S/SAI block diagram

48.2.3 Modes of operation

The module operates in these power modes: Run mode, stop modes, and Debug mode.

48.2.3.1 Run mode

In Run mode, the SAI transmitter and receiver operate normally.

48.2.3.2 Stop modes

In Stop mode, the transmitter is disabled after completing the current transmit frame, and, the receiver is disabled after completing the current receive frame. Entry into Stop mode is prevented–not acknowledged–while waiting for the transmitter and receiver to be disabled at the end of the current frame.

48.2.3.3 Debug mode

In Debug mode, the SAI transmitter and/or receiver can continue operating provided the Debug Enable bit is set. When TCSR[DBGE] or RCSR[DBGE] bit is clear and Debug mode is entered, the SAI is disabled after completing the current transmit or receive frame. The transmitter and receiver bit clocks are not affected by Debug mode.

48.3 External signals

Name	Function	I/O
SAI_TX_BCLK	Transmit Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	I/O
SAI_TX_SYNC	Transmit Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	I/O
SAI_TX_DATA[3:0]	Transmit Data. The transmit data is generated synchronously by the bit clock and is tristated whenever not transmitting a word.	O
SAI_RX_BCLK	Receive Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	I/O
SAI_RX_SYNC	Receive Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	I/O
SAI_RX_DATA[3:0]	Receive Data. The receive data is sampled synchronously by the bit clock.	I
SAI_MCLK	Audio Master Clock. The master clock is an input when externally generated and an output when internally generated.	I/O

48.4 Memory map and register definition

A read or write access to an address from offset 0x108 and above will result in a bus error.

I2S memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SAI Transmit Control Register (I2S_TCSR)	32	R/W	0000_0000h	48.4.1/2091
4	SAI Transmit Configuration 1 Register (I2S_TCR1)	32	R/W	0000_0000h	48.4.2/2094
8	SAI Transmit Configuration 2 Register (I2S_TCR2)	32	R/W	0000_0000h	48.4.3/2094
C	SAI Transmit Configuration 3 Register (I2S_TCR3)	32	R/W	0000_0000h	48.4.4/2096
10	SAI Transmit Configuration 4 Register (I2S_TCR4)	32	R/W	0000_0000h	48.4.5/2097
14	SAI Transmit Configuration 5 Register (I2S_TCR5)	32	R/W	0000_0000h	48.4.6/2099
20	SAI Transmit Data Register (I2S_TDR0)	32	W (always reads 0)	0000_0000h	48.4.7/2100
24	SAI Transmit Data Register (I2S_TDR1)	32	W (always reads 0)	0000_0000h	48.4.7/2100
28	SAI Transmit Data Register (I2S_TDR2)	32	W (always reads 0)	0000_0000h	48.4.7/2100
2C	SAI Transmit Data Register (I2S_TDR3)	32	W (always reads 0)	0000_0000h	48.4.7/2100
40	SAI Transmit FIFO Register (I2S_TFR0)	32	R	0000_0000h	48.4.8/2101
44	SAI Transmit FIFO Register (I2S_TFR1)	32	R	0000_0000h	48.4.8/2101
48	SAI Transmit FIFO Register (I2S_TFR2)	32	R	0000_0000h	48.4.8/2101
4C	SAI Transmit FIFO Register (I2S_TFR3)	32	R	0000_0000h	48.4.8/2101
60	SAI Transmit Mask Register (I2S_TMR)	32	R/W	0000_0000h	48.4.9/2101
80	SAI Receive Control Register (I2S_RCSR)	32	R/W	0000_0000h	48.4.10/ 2102
84	SAI Receive Configuration 1 Register (I2S_RCR1)	32	R/W	0000_0000h	48.4.11/ 2105
88	SAI Receive Configuration 2 Register (I2S_RCR2)	32	R/W	0000_0000h	48.4.12/ 2106
8C	SAI Receive Configuration 3 Register (I2S_RCR3)	32	R/W	0000_0000h	48.4.13/ 2107
90	SAI Receive Configuration 4 Register (I2S_RCR4)	32	R/W	0000_0000h	48.4.14/ 2108
94	SAI Receive Configuration 5 Register (I2S_RCR5)	32	R/W	0000_0000h	48.4.15/ 2110

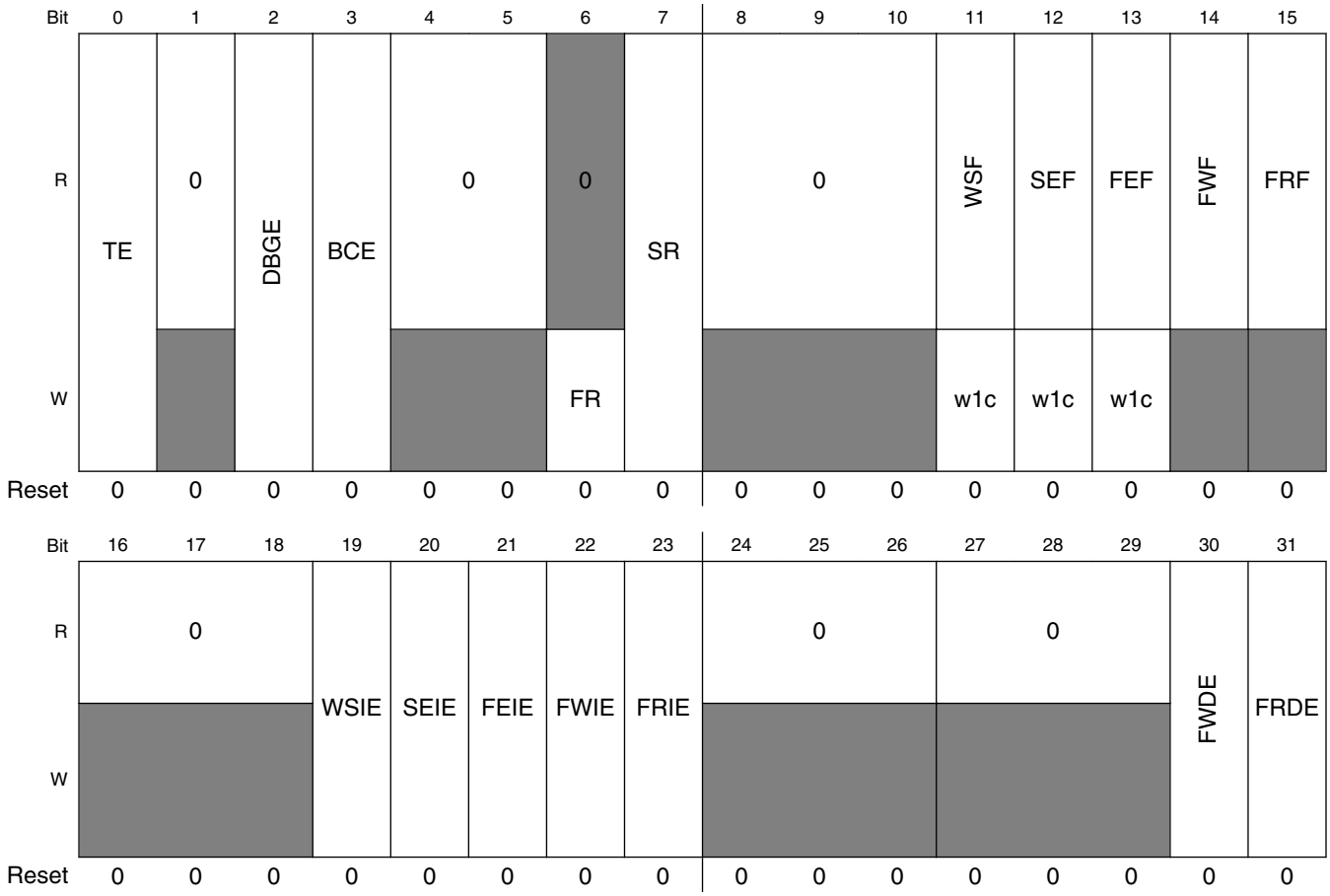
Table continues on the next page...

I2S memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
A0	SAI Receive Data Register (I2S_RDR0)	32	R	0000_0000h	48.4.16/ 2111
A4	SAI Receive Data Register (I2S_RDR1)	32	R	0000_0000h	48.4.16/ 2111
A8	SAI Receive Data Register (I2S_RDR2)	32	R	0000_0000h	48.4.16/ 2111
AC	SAI Receive Data Register (I2S_RDR3)	32	R	0000_0000h	48.4.16/ 2111
C0	SAI Receive FIFO Register (I2S_RFR0)	32	R	0000_0000h	48.4.17/ 2112
C4	SAI Receive FIFO Register (I2S_RFR1)	32	R	0000_0000h	48.4.17/ 2112
C8	SAI Receive FIFO Register (I2S_RFR2)	32	R	0000_0000h	48.4.17/ 2112
CC	SAI Receive FIFO Register (I2S_RFR3)	32	R	0000_0000h	48.4.17/ 2112
E0	SAI Receive Mask Register (I2S_RMR)	32	R/W	0000_0000h	48.4.18/ 2112
100	SAI MCLK Control Register (I2S_MCR)	32	R/W	0000_0000h	48.4.19/ 2113
104	SAI MCLK Divide Register (I2S_MDR)	32	R/W	0000_0000h	48.4.20/ 2114

48.4.1 SAI Transmit Control Register (I2S_TCSR)

Address: 0h base + 0h offset = 0h



I2S_TCSR field descriptions

Field	Description
0 TE	Transmitter Enable Enables/disables the transmitter. When software clears this field, the transmitter remains enabled, and this bit remains set, until the end of the current frame. 0 Transmitter is disabled. 1 Transmitter is enabled, or transmitter has been disabled and has not yet reached end of frame.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 DBGE	Debug Enable Enables/disables transmitter operation in Debug mode. The transmit bit clock is not affected by debug mode. 0 Transmitter is disabled in Debug mode, after completing the current frame. 1 Transmitter is enabled in Debug mode.

Table continues on the next page...

I2S_TCSR field descriptions (continued)

Field	Description
3 BCE	Bit Clock Enable Enables the transmit bit clock, separately from the TE. This field is automatically set whenever TE is set. When software clears this field, the transmit bit clock remains enabled, and this bit remains set, until the end of the current frame. 0 Transmit bit clock is disabled. 1 Transmit bit clock is enabled.
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 FR	FIFO Reset Resets the FIFO pointers. Reading this field will always return zero. FIFO pointers should only be reset when the transmitter is disabled or the FIFO error flag is set. 0 No effect. 1 FIFO reset.
7 SR	Software Reset When set, resets the internal transmitter logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers. 0 No effect. 1 Software reset.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 WSF	Word Start Flag Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag. 0 Start of word not detected. 1 Start of word detected.
12 SEF	Sync Error Flag Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag. 0 Sync error not detected. 1 Frame sync error detected.
13 FEF	FIFO Error Flag Indicates that an enabled transmit FIFO has underrun. Write a logic 1 to this field to clear this flag. 0 Transmit underrun not detected. 1 Transmit underrun detected.
14 FWF	FIFO Warning Flag Indicates that an enabled transmit FIFO is empty. 0 No enabled transmit FIFO is empty. 1 Enabled transmit FIFO is empty.

Table continues on the next page...

I2S_TCSR field descriptions (continued)

Field	Description
15 FRF	FIFO Request Flag Indicates that the number of words in an enabled transmit channel FIFO is less than or equal to the transmit FIFO watermark. 0 Transmit FIFO watermark has not been reached. 1 Transmit FIFO watermark has been reached.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 WSIE	Word Start Interrupt Enable Enables/disables word start interrupts. 0 Disables interrupt. 1 Enables interrupt.
20 SEIE	Sync Error Interrupt Enable Enables/disables sync error interrupts. 0 Disables interrupt. 1 Enables interrupt.
21 FEIE	FIFO Error Interrupt Enable Enables/disables FIFO error interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
22 FWIE	FIFO Warning Interrupt Enable Enables/disables FIFO warning interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
23 FRIE	FIFO Request Interrupt Enable Enables/disables FIFO request interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 FWDE	FIFO Warning DMA Enable Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.
31 FRDE	FIFO Request DMA Enable

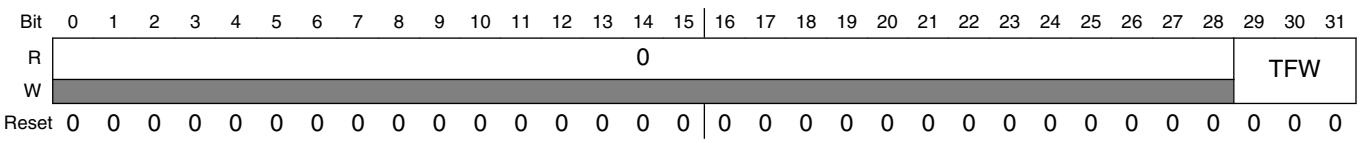
Table continues on the next page...

I2S_TCSR field descriptions (continued)

Field	Description
	Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.

48.4.2 SAI Transmit Configuration 1 Register (I2S_TCR1)

Address: 0h base + 4h offset = 4h



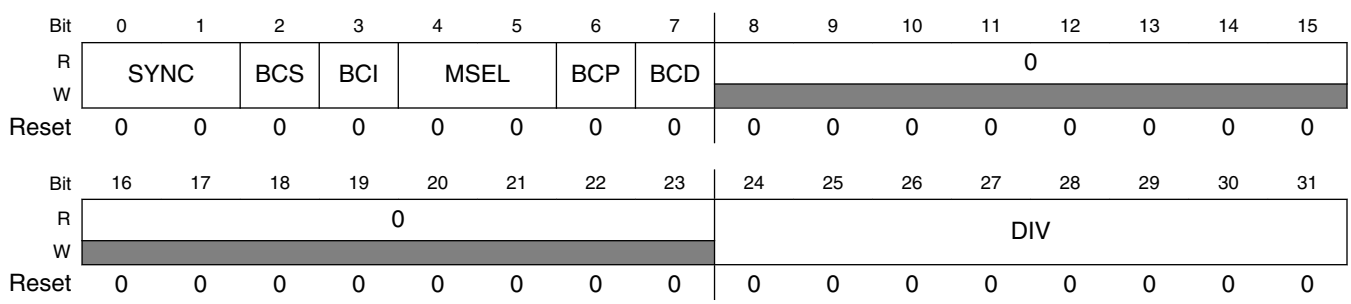
I2S_TCR1 field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TFW	Transmit FIFO Watermark Configures the watermark level for all enabled transmit channels.

48.4.3 SAI Transmit Configuration 2 Register (I2S_TCR2)

This register must not be altered when TCSR[TE] is set.

Address: 0h base + 8h offset = 8h



I2S_TCR2 field descriptions

Field	Description
0–1 SYNC	Synchronous Mode

Table continues on the next page...

I2S_TCR2 field descriptions (continued)

Field	Description
	Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the receiver or other SAI peripheral must be configured for asynchronous operation. 00 Asynchronous mode. 01 Synchronous with receiver. 10 Synchronous with another SAI transmitter. 11 Synchronous with another SAI receiver.
2 BCS	Bit Clock Swap This field swaps the bit clock used by the transmitter. When the transmitter is configured in asynchronous mode and this bit is set, the transmitter is clocked by the receiver bit clock (SAI_RX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the transmitter continues to use the transmit frame sync (SAI_TX_SYNC). When the transmitter is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the transmitter bit clock (SAI_TX_BCLK) but use the receiver frame sync (SAI_RX_SYNC). This field has no effect when synchronous to another SAI peripheral. 0 Use the normal bit clock source. 1 Swap the bit clock source.
3 BCI	Bit Clock Input When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the transmitter is delayed by the pad output delay (the transmitter is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time. The slave mode timing from the datasheet should be used for the transmitter when this bit is set. In synchronous mode, this bit allows the transmitter to use the slave mode timing from the datasheet, while the receiver uses the master mode timing. This field has no effect when configured for an externally generated bit clock or when synchronous to another SAI peripheral . 0 No effect. 1 Internal logic is clocked as if bit clock was externally generated.
4–5 MSEL	MCLK Select Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock. **NOTE:** Depending on the device, some Master Clock options might not be available. See the chip-specific information for the meaning of each option. 00 Bus Clock selected. 01 Master Clock (MCLK) 1 option selected. 10 Master Clock (MCLK) 2 option selected. 11 Master Clock (MCLK) 3 option selected.
6 BCP	Bit Clock Polarity Configures the polarity of the bit clock. 0 Bit clock is active high with drive outputs on rising edge and sample inputs on falling edge. 1 Bit clock is active low with drive outputs on falling edge and sample inputs on rising edge.

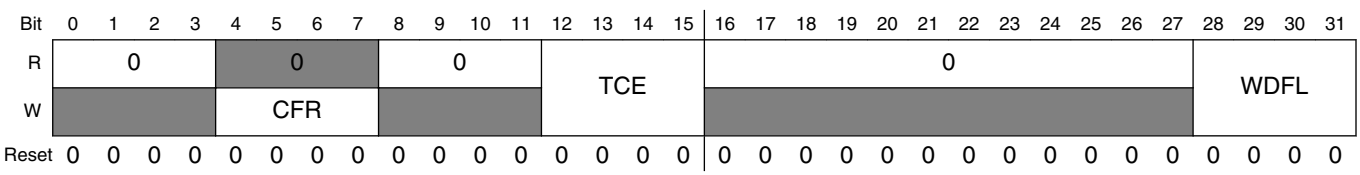
Table continues on the next page...

I2S_TCR2 field descriptions (continued)

Field	Description
7 BCD	Bit Clock Direction Configures the direction of the bit clock. 0 Bit clock is generated externally in Slave mode. 1 Bit clock is generated internally in Master mode.
8–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 DIV	Bit Clock Divide Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2.

48.4.4 SAI Transmit Configuration 3 Register (I2S_TCR3)

Address: 0h base + Ch offset = Ch



I2S_TCR3 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 CFR	Channel FIFO Reset Resets the FIFO pointers for a specific channel. Reading this field will always return zero. FIFO pointers should only be reset when a channel is disabled or the FIFO error flag is set. 0 No effect. 1 Transmit data channel N FIFO is reset.
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 TCE	Transmit Channel Enable Enables the corresponding data channel for transmit operation. A channel must be enabled before its FIFO is accessed. Changing this field will take effect immediately for generating the FIFO request and warning flags, but at the end of each frame for transmit operation. 0 Transmit data channel N is disabled. 1 Transmit data channel N is enabled.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

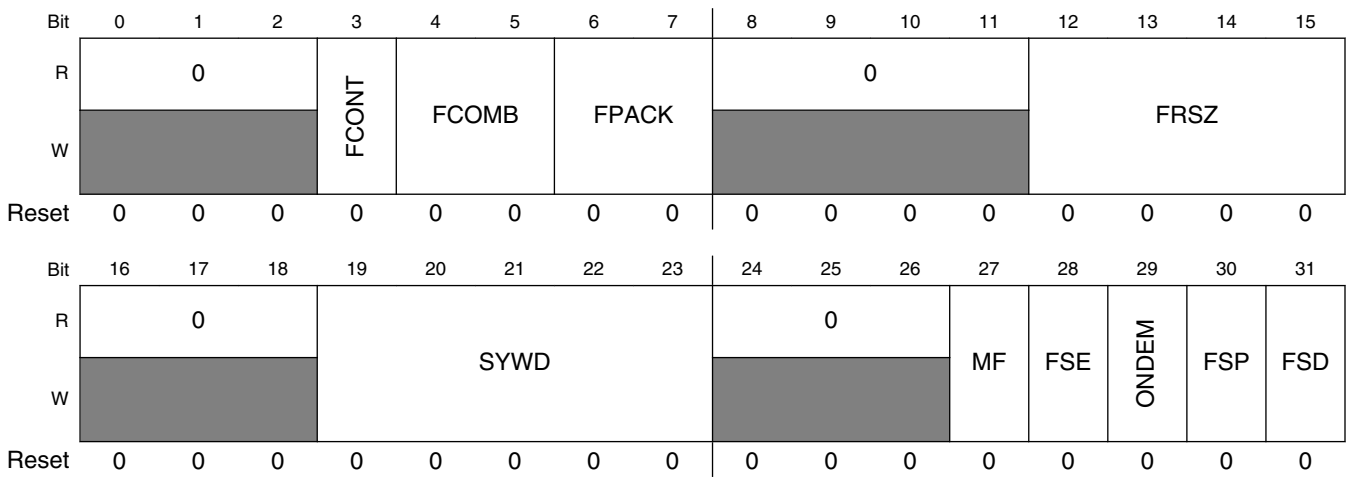
I2S_TCR3 field descriptions (continued)

Field	Description
28–31 WDFL	Word Flag Configuration Configures which word sets the start of word flag. The value written must be one less than the word number. For example, writing 0 configures the first word in the frame. When configured to a value greater than TCR4[FRSZ], then the start of word flag is never set.

48.4.5 SAI Transmit Configuration 4 Register (I2S_TCR4)

This register must not be altered when TCSR[TE] is set.

Address: 0h base + 10h offset = 10h



I2S_TCR4 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FCONT	FIFO Continue on Error Configures when the SAI will continue transmitting after a FIFO error has been detected. 0 On FIFO error, the SAI will continue from the start of the next frame after the FIFO error flag has been cleared. 1 On FIFO error, the SAI will continue from the same word that caused the FIFO error to set after the FIFO warning flag has been cleared.
4–5 FCOMB	FIFO Combine Mode When FIFO combine mode is enabled for FIFO writes, software writing to any FIFO data register will alternate the write among the enabled data channel FIFOs. For example, if two data channels are enabled then the first write will be performed to the first enabled data channel FIFO and the second write will be performed to the second enabled data channel FIFO. Reseting the FIFO or disabling FIFO combine mode for FIFO writes will reset the pointer back to the first enabled data channel.

Table continues on the next page...

I2S_TCR4 field descriptions (continued)

Field	Description
	When FIFO combine mode is enabled for FIFO reads from the transmit shift registers, the transmit data channel output will alternate between the enabled data channel FIFOs. For example, if two data channels are enabled then the first unmasked word will be transmitted from the first enabled data channel FIFO and the second unmasked word will be transmitted from the second enabled data channel FIFO. Since the first word of the frame is always transmitted from the first enabled data channel FIFO, it is recommended that the number of unmasked words per frame is evenly divisible by the number of enabled data channels. 00 FIFO combine mode disabled. 01 FIFO combine mode enabled on FIFO reads (from transmit shift registers). 10 FIFO combine mode enabled on FIFO writes (by software). 11 FIFO combine mode enabled on FIFO reads (from transmit shift registers) and writes (by software).
6–7 FPACK	FIFO Packing Mode Enables packing of 8-bit data or 16-bit data into each 32-bit FIFO word. If the word size is greater than 8-bit or 16-bit then only the first 8-bit or 16-bits are loaded from the FIFO. The first word in each frame always starts with a new 32-bit FIFO word and the first bit shifted must be configured within the first packed word. When FIFO packing is enabled, the FIFO write pointer will only increment when the full 32-bit FIFO word has been written by software. 00 FIFO packing is disabled 01 Reserved 10 8-bit FIFO packing is enabled 11 16-bit FIFO packing is enabled
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 FRSZ	Frame size Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 16 words.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 SYWD	Sync Width Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 MF	MSB First Configures whether the LSB or the MSB is transmitted first. 0 LSB is transmitted first. 1 MSB is transmitted first.
28 FSE	Frame Sync Early 0 Frame sync asserts with the first bit of the frame. 1 Frame sync asserts one bit before the first bit of the frame.
29 ONDEM	On Demand Mode

Table continues on the next page...

I2S_TCR4 field descriptions (continued)

Field	Description
	When set, and the frame sync is generated internally, a frame sync is only generated when the FIFO warning flag is clear. 0 Internal frame sync is generated continuously. 1 Internal frame sync is generated when the FIFO warning flag is clear.
30 FSP	Frame Sync Polarity Configures the polarity of the frame sync. 0 Frame sync is active high. 1 Frame sync is active low.
31 FSD	Frame Sync Direction Configures the direction of the frame sync. 0 Frame sync is generated externally in Slave mode. 1 Frame sync is generated internally in Master mode.

48.4.6 SAI Transmit Configuration 5 Register (I2S_TCR5)

This register must not be altered when TCSR[TE] is set.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			WNW					0			W0W					0			FBT					0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2S_TCR5 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 WNW	Word N Width Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 W0W	Word 0 Width Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

I2S_TCR5 field descriptions (continued)

Field	Description
19–23 FBT	First Bit Shifted Configures the bit index for the first bit transmitted for each word in the frame. If configured for MSB First, the index of the next bit transmitted is one less than the current bit transmitted. If configured for LSB First, the index of the next bit transmitted is one more than the current bit transmitted. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.7 SAI Transmit Data Register (I2S_TDR*n*)

Address: 0h base + 20h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W	TDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2S_TDR*n* field descriptions

Field	Description
0–31 TDR	Transmit Data Register The corresponding TCR3[TCE] bit must be set before accessing the channel's transmit data register. Writes to this register when the transmit FIFO is not full will push the data written into the transmit data FIFO. Writes to this register when the transmit FIFO is full are ignored.

48.4.8 SAI Transmit FIFO Register (I2S_TFR*n*)

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

Address: 0h base + 40h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WCP	0											WFP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												RFP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2S_TFR*n* field descriptions

Field	Description
0 WCP	Write Channel Pointer When FIFO Combine mode is enabled for writes, indicates that this data channel is the next FIFO to be written. 0 No effect. 1 FIFO combine is enabled for FIFO writes and this FIFO will be written on the next FIFO write.
1–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 WFP	Write FIFO Pointer FIFO write pointer for transmit data channel.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 RFP	Read FIFO Pointer FIFO read pointer for transmit data channel.

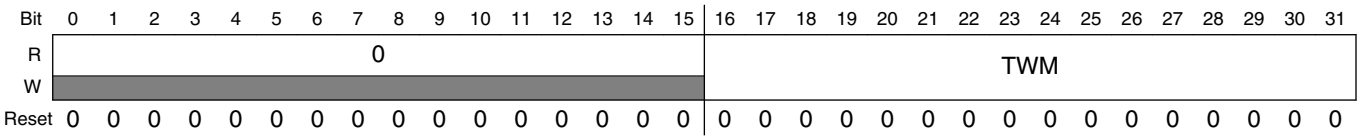
48.4.9 SAI Transmit Mask Register (I2S_TMR)

This register is double-buffered and updates:

1. When TCSR[TE] is first set
2. At the end of each frame.

This allows the masked words in each frame to change from frame to frame.

Address: 0h base + 60h offset = 60h

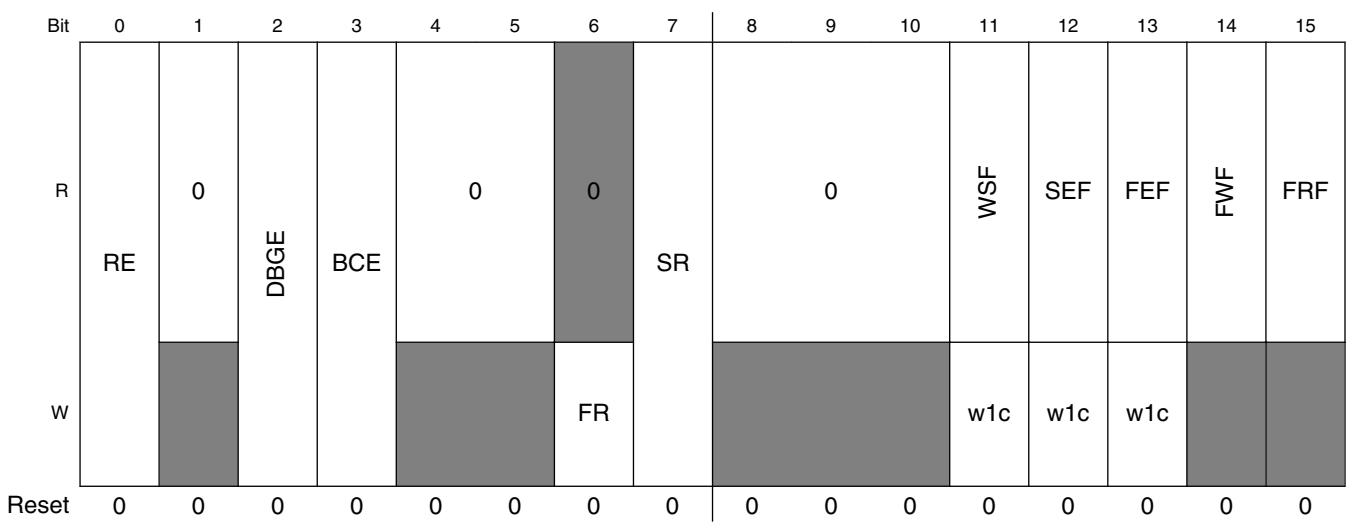


I2S_TMR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 TWM	Transmit Word Mask Configures whether the transmit word is masked (transmit data pin tristated and transmit data not read from FIFO) for the corresponding word in the frame. 0 Word N is enabled. 1 Word N is masked. The transmit data pins are tri-stated when masked.

48.4.10 SAI Receive Control Register (I2S_RCSR)

Address: 0h base + 80h offset = 80h



I2S_RCSR field descriptions

Field	Description
0 RE	Receiver Enable Enables/disables the receiver. When software clears this field, the receiver remains enabled, and this bit remains set, until the end of the current frame. 0 Receiver is disabled. 1 Receiver is enabled, or receiver has been disabled and has not yet reached end of frame.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 DBGE	Debug Enable Enables/disables receiver operation in Debug mode. The receive bit clock is not affected by Debug mode. 0 Receiver is disabled in Debug mode, after completing the current frame. 1 Receiver is enabled in Debug mode.
3 BCE	Bit Clock Enable Enables the receive bit clock, separately from RE. This field is automatically set whenever RE is set. When software clears this field, the receive bit clock remains enabled, and this field remains set, until the end of the current frame. 0 Receive bit clock is disabled. 1 Receive bit clock is enabled.
4–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 FR	FIFO Reset Resets the FIFO pointers. Reading this field will always return zero. FIFO pointers should only be reset when the receiver is disabled or the FIFO error flag is set. 0 No effect. 1 FIFO reset.
7 SR	Software Reset Resets the internal receiver logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers. 0 No effect. 1 Software reset.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 WSF	Word Start Flag Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag. 0 Start of word not detected. 1 Start of word detected.
12 SEF	Sync Error Flag Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag.

Table continues on the next page...

I2S_RCSR field descriptions (continued)

Field	Description
	0 Sync error not detected. 1 Frame sync error detected.
13 FEF	FIFO Error Flag Indicates that an enabled receive FIFO has overflowed. Write a logic 1 to this field to clear this flag. 0 Receive overflow not detected. 1 Receive overflow detected.
14 FWF	FIFO Warning Flag Indicates that an enabled receive FIFO is full. 0 No enabled receive FIFO is full. 1 Enabled receive FIFO is full.
15 FRF	FIFO Request Flag Indicates that the number of words in an enabled receive channel FIFO is greater than the receive FIFO watermark. 0 Receive FIFO watermark not reached. 1 Receive FIFO watermark has been reached.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 WSIE	Word Start Interrupt Enable Enables/disables word start interrupts. 0 Disables interrupt. 1 Enables interrupt.
20 SEIE	Sync Error Interrupt Enable Enables/disables sync error interrupts. 0 Disables interrupt. 1 Enables interrupt.
21 FEIE	FIFO Error Interrupt Enable Enables/disables FIFO error interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
22 FWIE	FIFO Warning Interrupt Enable Enables/disables FIFO warning interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
23 FRIE	FIFO Request Interrupt Enable Enables/disables FIFO request interrupts.

Table continues on the next page...

I2S_RCSR field descriptions (continued)

Field	Description
	0 Disables the interrupt. 1 Enables the interrupt.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 FWDE	FIFO Warning DMA Enable Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.
31 FRDE	FIFO Request DMA Enable Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.

48.4.11 SAI Receive Configuration 1 Register (I2S_RCR1)

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																													RFW		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2S_RCR1 field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 RFW	Receive FIFO Watermark Configures the watermark level for all enabled receiver channels.

48.4.12 SAI Receive Configuration 2 Register (I2S_RCR2)

This register must not be altered when RCSR[RE] is set.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SYNC		BCS	BCI	MSEL		BCP	BCD	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								DIV							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2S_RCR2 field descriptions

Field	Description
0–1 SYNC	Synchronous Mode Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the transmitter or other SAI peripheral must be configured for asynchronous operation. 00 Asynchronous mode. 01 Synchronous with transmitter. 10 Synchronous with another SAI receiver. 11 Synchronous with another SAI transmitter.
2 BCS	Bit Clock Swap This field swaps the bit clock used by the receiver. When the receiver is configured in asynchronous mode and this bit is set, the receiver is clocked by the transmitter bit clock (SAI_TX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the receiver continues to use the receiver frame sync (SAI_RX_SYNC). When the receiver is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the receiver bit clock (SAI_RX_BCLK) but use the transmitter frame sync (SAI_TX_SYNC). This field has no effect when synchronous to another SAI peripheral. 0 Use the normal bit clock source. 1 Swap the bit clock source.
3 BCI	Bit Clock Input When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the receiver is delayed by the pad output delay (the receiver is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time. The slave mode timing from the datasheet should be used for the receiver when this bit is set. In synchronous mode, this bit allows the receiver to use the slave mode timing from the datasheet, while the transmitter uses the master mode timing. This field has no effect when configured for an externally generated bit clock or when synchronous to another SAI peripheral .

Table continues on the next page...

I2S_RCR2 field descriptions (continued)

Field	Description
	0 No effect. 1 Internal logic is clocked as if bit clock was externally generated.
4–5 MSEL	MCLK Select Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock. **NOTE:** Depending on the device, some Master Clock options might not be available. See the chip-specific information for the availability and chip-specific meaning of each option. 00 Bus Clock selected. 01 Master Clock (MCLK) 1 option selected. 10 Master Clock (MCLK) 2 option selected. 11 Master Clock (MCLK) 3 option selected.
6 BCP	Bit Clock Polarity Configures the polarity of the bit clock. 0 Bit Clock is active high with drive outputs on rising edge and sample inputs on falling edge. 1 Bit Clock is active low with drive outputs on falling edge and sample inputs on rising edge.
7 BCD	Bit Clock Direction Configures the direction of the bit clock. 0 Bit clock is generated externally in Slave mode. 1 Bit clock is generated internally in Master mode.
8–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 DIV	Bit Clock Divide Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2.

48.4.13 SAI Receive Configuration 3 Register (I2S_RCR3)

Address: 0h base + 8Ch offset = 8Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				0				0				RCE				0												WDFL			
W					CFR																											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2S_RCR3 field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

I2S_RCR3 field descriptions (continued)

Field	Description
4–7 CFR	Channel FIFO Reset Resets the FIFO pointers for a specific channel. Reading this field will always return zero. FIFO pointers should only be reset when a channel is disabled or the FIFO error flag is set. 0 No effect. 1 Receive data channel N FIFO is reset.
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 RCE	Receive Channel Enable Enables the corresponding data channel for receive operation. A channel must be enabled before its FIFO is accessed. Changing this field will take effect immediately for generating the FIFO request and warning flags, but at the end of each frame for receive operation. 0 Receive data channel N is disabled. 1 Receive data channel N is enabled.
16–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 WDFL	Word Flag Configuration Configures which word the start of word flag is set. The value written should be one less than the word number (for example, write zero to configure for the first word in the frame). When configured to a value greater than the Frame Size field, then the start of word flag is never set.

48.4.14 SAI Receive Configuration 4 Register (I2S_RCR4)

This register must not be altered when RCSR[RE] is set.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0			FCONT	FCOMB		FPACK		0				FRSZ			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			SYWD					0			MF	FSE	ONDEM	FSP	FSD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2S_RCR4 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 FCONT	FIFO Continue on Error Configures when the SAI will continue receiving after a FIFO error has been detected. 0 On FIFO error, the SAI will continue from the start of the next frame after the FIFO error flag has been cleared. 1 On FIFO error, the SAI will continue from the same word that caused the FIFO error to set after the FIFO warning flag has been cleared.
4–5 FCOMB	FIFO Combine Mode When FIFO combine mode is enabled for FIFO reads, software reading any FIFO data register will alternate the read among the enabled data channel FIFOs. For example, if two data channels are enabled then the first read will be performed to the first enabled data channel FIFO and the second read will be performed to the second enabled data channel FIFO. Reseting the FIFO or disabling FIFO combine mode for FIFO reads will reset the pointer back to the first enabled data channel. When FIFO combine mode is enabled for FIFO writes from the receive shift registers, the first enabled data channel input will alternate between the enabled data channel FIFOs. For example, if two data channels are enabled then the first unmasked received word will be stored in the first enabled data channel FIFO and the second unmasked received word will be stored in the second enabled data channel FIFO. Since the first word of the frame is always stored in the first enabled data channel FIFO, it is recommended that the number of unmasked words per frame is evenly divisible by the number of enabled data channels. 00 FIFO combine mode disabled. 01 FIFO combine mode enabled on FIFO writes (from receive shift registers). 10 FIFO combine mode enabled on FIFO reads (by software). 11 FIFO combine mode enabled on FIFO writes (from receive shift registers) and reads (by software).
6–7 FPACK	FIFO Packing Mode Enables packing of 8-bit data or 16-bit data into each 32-bit FIFO word. If the word size is greater than 8-bit or 16-bit then only the first 8-bit or 16-bits are stored to the FIFO. The first word in each frame always starts with a new 32-bit FIFO word and the first bit shifted must be configured within the first packed word. When FIFO packing is enabled, the FIFO read pointer will only increment when the full 32-bit FIFO word has been read by software. 00 FIFO packing is disabled 01 Reserved. 10 8-bit FIFO packing is enabled 11 16-bit FIFO packing is enabled
8–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 FRSZ	Frame Size Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 16 words.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 SYWD	Sync Width

Table continues on the next page...

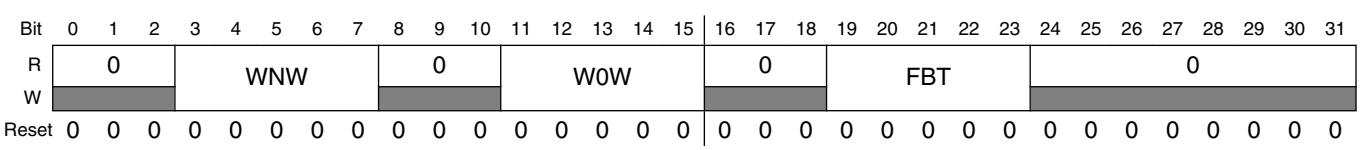
I2S_RCR4 field descriptions (continued)

Field	Description
	Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 MF	MSB First Configures whether the LSB or the MSB is received first. 0 LSB is received first. 1 MSB is received first.
28 FSE	Frame Sync Early 0 Frame sync asserts with the first bit of the frame. 1 Frame sync asserts one bit before the first bit of the frame.
29 ONDEM	On Demand Mode When set, and the frame sync is generated internally, a frame sync is only generated when the FIFO warning flag is clear. 0 Internal frame sync is generated continuously. 1 Internal frame sync is generated when the FIFO warning flag is clear.
30 FSP	Frame Sync Polarity Configures the polarity of the frame sync. 0 Frame sync is active high. 1 Frame sync is active low.
31 FSD	Frame Sync Direction Configures the direction of the frame sync. 0 Frame Sync is generated externally in Slave mode. 1 Frame Sync is generated internally in Master mode.

48.4.15 SAI Receive Configuration 5 Register (I2S_RCR5)

This register must not be altered when RCSR[RE] is set.

Address: 0h base + 94h offset = 94h



I2S_RCR5 field descriptions

Field	Description
0–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 WNW	Word N Width Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–15 W0W	Word 0 Width Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 FBT	First Bit Shifted Configures the bit index for the first bit received for each word in the frame. If configured for MSB First, the index of the next bit received is one less than the current bit received. If configured for LSB First, the index of the next bit received is one more than the current bit received. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First.
24–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.16 SAI Receive Data Register (I2S_RDR*n*)

Reading this register introduces one additional peripheral clock wait state on each read.

Address: 0h base + A0h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2S_RDR*n* field descriptions

Field	Description
0–31 RDR	Receive Data Register The corresponding RCR3[RCE] bit must be set before accessing the channel's receive data register. Reads from this register when the receive FIFO is not empty will return the data from the top of the receive FIFO. Reads from this register when the receive FIFO is empty are ignored.

48.4.17 SAI Receive FIFO Register (I2S_RFR*n*)

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

Address: 0h base + C0h offset + (4d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0												WFP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	RCP	0											RFP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2S_RFR*n* field descriptions

Field	Description
0–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 WFP	Write FIFO Pointer FIFO write pointer for receive data channel.
16 RCP	Receive Channel Pointer When FIFO Combine mode is enabled for reads, indicates that this data channel is the next FIFO to be read. 0 No effect. 1 FIFO combine is enabled for FIFO reads and this FIFO will be read on the next FIFO read.
17–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 RFP	Read FIFO Pointer FIFO read pointer for receive data channel.

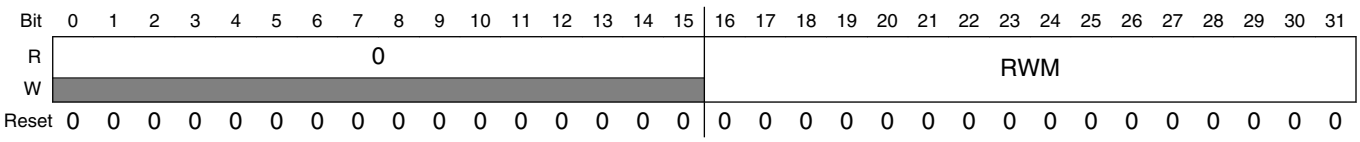
48.4.18 SAI Receive Mask Register (I2S_RMR)

This register is double-buffered and updates:

1. When RCSR[RE] is first set
2. At the end of each frame

This allows the masked words in each frame to change from frame to frame.

Address: 0h base + E0h offset = E0h



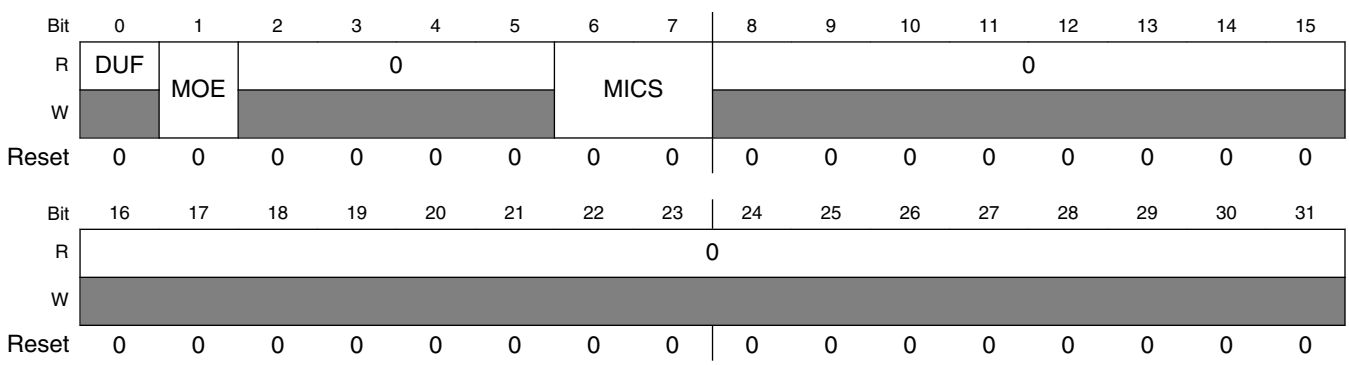
I2S_RMR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 RWM	Receive Word Mask Configures whether the receive word is masked (received data ignored and not written to receive FIFO) for the corresponding word in the frame. 0 Word N is enabled. 1 Word N is masked.

48.4.19 SAI MCLK Control Register (I2S_MCR)

The MCLK Control Register (MCR) controls the clock source and direction of the audio master clock.

Address: 0h base + 100h offset = 100h



I2S_MCR field descriptions

Field	Description
0 DUF	Divider Update Flag Provides the status of on-the-fly updates to the MCLK divider ratio. 0 MCLK divider ratio is not being updated currently. 1 MCLK divider ratio is updating on-the-fly. Further updates to the MCLK divider ratio are blocked while this flag remains set.
1 MOE	MCLK Output Enable

Table continues on the next page...

I2S_MCR field descriptions (continued)

Field	Description
	Enables the MCLK divider and configures the MCLK signal pin as an output. When software clears this field, it remains set until the MCLK divider is fully disabled. 0 MCLK signal pin is configured as an input that bypasses the MCLK divider. 1 MCLK signal pin is configured as an output from the MCLK divider and the MCLK divider is enabled.
2–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–7 MICS	MCLK Input Clock Select Selects the clock input to the MCLK divider. This field cannot be changed while the MCLK divider is enabled. See the chip-specific information for the connections to these inputs. 00 MCLK divider input clock 0 is selected. 01 Reserved 10 MCLK divider input clock 2 is selected. 11 MCLK divider input clock 3 is selected.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

48.4.20 SAI MCLK Divide Register (I2S_MDR)

The MCLK Divide Register (MDR) configures the MCLK divide ratio. Although the MDR can be changed when the MCLK divider clock is enabled, additional writes to the MDR are blocked while MCR[DUF] is set. Writes to the MDR when the MCLK divided clock is disabled do not set MCR[DUF].

Address: 0h base + 104h offset = 104h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											FRACT					DIVIDE															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2S_MDR field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11–19 FRACT	MCLK Fraction Sets the MCLK divide ratio such that: MCLK output = MCLK input * ((FRACT + 1) / (DIVIDE + 1)). FRACT must be set equal or less than the value in the DIVIDE field. **NOTE:** When using fractional divide values, the MCLK duty cycle will not always be 50/50. See Audio master clock.
20–31 DIVIDE	MCLK Divide

Table continues on the next page...

I2S_MDR field descriptions (continued)

Field	Description
	Sets the MCLK divide ratio such that: MCLK output = MCLK input * ((FRACT + 1) / (DIVIDE + 1)). FRACT must be set equal or less than the value in the DIVIDE field. **NOTE:** When using fractional divide values, the MCLK duty cycle will not always be 50/50. See Audio master clock.

48.5 Functional description

This section provides a complete functional description of the block.

48.5.1 SAI clocking

The SAI clocks include:

- The audio master clock
- The bit clock
- The bus clock

48.5.1.1 Audio master clock

The audio master clock is used to generate the bit clock when the receiver or transmitter is configured for an internally generated bit clock. The transmitter and receiver can independently select between the bus clock and up to three audio master clocks to generate the bit clock.

Each SAI peripheral can control the input clock selection, pin direction and divide ratio of one audio master clock. The input clock selection and pin direction cannot be altered if an SAI module using that audio master clock has been enabled. The MCLK divide ratio can be altered while an SAI is using that master clock, although the change in the divide ratio takes several cycles. MCR[DUF] can be polled to determine when the divide ratio change has completed.

The audio master clock generation and selection is chip-specific. Refer to chip-specific clocking information about how the audio master clocks are generated. A typical implementation appears in the following figure.

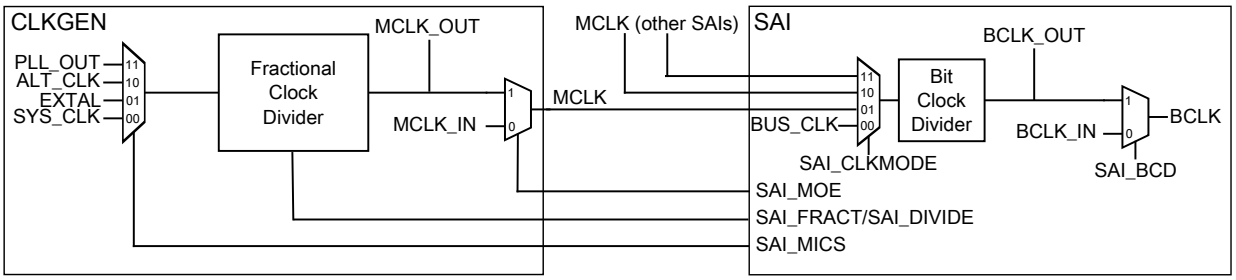


Figure 48-2. SAI master clock generation

The MCLK fractional clock divider uses both clock edges from the input clock to generate a divided down clock that will approximate the output frequency, but without creating any new clock edges. Configuring FRACT and DIVIDE to the same value will result in a divide by 1 clock, while configuring FRACT higher than DIVIDE is not supported. The duty cycle can range from 66/33 when FRACT is set to one less than DIVIDE down to 50/50 for integer divide ratios, and will approach 50/50 for large non-integer divide ratios. There is no cycle to cycle jitter or duty cycle variance when the divide ratio is an integer or half integer, otherwise the divider output will oscillate between the two divided frequencies that are the closest integer or half integer divisors of the divider input clock frequency. The maximum jitter is therefore equal to half the divider input clock period, since both edges of the input clock are used in generating the divided clock.

48.5.1.2 Bit clock

The SAI transmitter and receiver support asynchronous free-running bit clocks that can be generated internally from an audio master clock or supplied externally. There is also the option for synchronous bit clock and frame sync operation between the receiver and transmitter or between multiple SAI peripherals.

Externally generated bit clocks must be:

- Enabled before the SAI transmitter or receiver is enabled
- Disabled after the SAI transmitter or receiver is disabled and completes its current frames

If the SAI transmitter or receiver is using an externally generated bit clock in asynchronous mode and that bit clock is generated by an SAI that is disabled in stop mode, then the transmitter or receiver should be disabled by software before entering stop mode. This issue does not apply when the transmitter or receiver is in a synchronous mode because all synchronous SAIs are enabled and disabled simultaneously.

48.5.1.3 Bus clock

The bus clock is used by the control and configuration registers and to generate synchronous interrupts and DMA requests.

NOTE

Although there is no specific minimum bus clock frequency specified, the bus clock frequency must be fast enough (relative to the bit clock frequency) to ensure that the FIFOs can be serviced, without generating either a transmitter FIFO underrun or receiver FIFO overflow condition.

48.5.2 SAI resets

The SAI is asynchronously reset on system reset. The SAI has a software reset and a FIFO reset.

48.5.2.1 Software reset

The SAI transmitter includes a software reset that resets all transmitter internal logic, including the bit clock generation, status flags, and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

The SAI receiver includes a software reset that resets all receiver internal logic, including the bit clock generation, status flags and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

48.5.2.2 FIFO reset

The SAI transmitter includes a FIFO reset that synchronizes the FIFO write pointer to the same value as the FIFO read pointer. This empties the FIFO contents and is to be used after TCSR[FEF] is set, and before the FIFO is re-initialized and TCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

The SAI transmitter can also reset the FIFO of individual data channels by setting the appropriate TCR3[CFR] bit. This should only be done when the corresponding TCR3[TCE] bit is clear.

The SAI receiver includes a FIFO reset that synchronizes the FIFO read pointer to the same value as the FIFO write pointer. This empties the FIFO contents and is to be used after the RCSR[FEF] is set and any remaining data has been read from the FIFO, and before the RCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

The SAI receiver can also reset the FIFO of individual data channels by setting the appropriate RCR3[CFR] bit. This should only be done when the corresponding RCR3[RCE] bit is clear.

48.5.3 Synchronous modes

The SAI transmitter and receiver can operate synchronously to each other or synchronously to other SAI peripherals.

48.5.3.1 Synchronous mode

The SAI transmitter and receiver can be configured to operate with synchronous bit clock and frame sync.

If the transmitter bit clock and frame sync are to be used by both the transmitter and receiver:

- The transmitter must be configured for asynchronous operation and the receiver for synchronous operation.
- In synchronous mode, the receiver is enabled only when both the transmitter and receiver are enabled.
- It is recommended that the transmitter is the last enabled and the first disabled.

If the receiver bit clock and frame sync are to be used by both the transmitter and receiver:

- The receiver must be configured for asynchronous operation and the transmitter for synchronous operation.
- In synchronous mode, the transmitter is enabled only when both the receiver and transmitter are both enabled.
- It is recommended that the receiver is the last enabled and the first disabled.

When operating in synchronous mode, only the bit clock, frame sync, and transmitter/receiver enable are shared. The transmitter and receiver otherwise operate independently, although configuration registers must be configured consistently across both the transmitter and receiver.

48.5.3.2 Multiple SAI Synchronous mode

Synchronous operation between multiple SAI peripherals is not supported on all devices. This mode requires the source of the bit clock and frame sync to be configured for asynchronous operation and the remaining users of the bit clock and frame sync to be configured for synchronous operation.

Synchronous operation between multiple SAI transmitters or receivers also requires the source of the bit clock and frame sync to be enabled for any of the synchronous transmitters or receivers to also be enabled. It is recommended that the source of the bit clock and frame sync is the last enabled and the first disabled.

When operating in synchronous mode, only the bit clock, frame sync, and transmitter/receiver enable are shared. The separate SAI peripherals otherwise operate independently, although configuration registers must be configured consistently across both the transmitter and receiver.

48.5.4 Frame sync configuration

When enabled, the SAI continuously transmits and/or receives frames of data. Each frame consists of a fixed number of words and each word consists of a fixed number of bits. Within each frame, any given word can be masked causing the receiver to ignore that word and the transmitter to tri-state for the duration of that word.

The frame sync signal is used to indicate the start of each frame. A valid frame sync requires a rising edge (if active high) or falling edge (if active low) to be detected and the transmitter or receiver cannot be busy with a previous frame. A valid frame sync is also ignored (slave mode) or not generated (master mode) for the first four bit clock cycles after enabling the transmitter or receiver.

The transmitter and receiver frame sync can be configured independently with any of the following options:

- Externally generated or internally generated
- Active high or active low
- Assert with the first bit in frame or asserts one bit early
- Assert for a duration between 1 bit clock and the first word length
- Frame length from 1 to 16 words per frame
- Word length to support 8 to 32 bits per word
    - First word length and remaining word lengths can be configured separately
- Words can be configured to transmit/receive MSB first or LSB first

These configuration options cannot be changed after the SAI transmitter or receiver is enabled.

48.5.5 Data FIFO

Each transmit and receive channel includes a FIFO of size 8 × 32-bit. The FIFO data is accessed using the SAI Transmit/Receive Data Registers.

48.5.5.1 Data alignment

Data in the FIFO can be aligned anywhere within the 32-bit wide register through the use of the First Bit Shifted configuration field, which selects the bit index (between 31 and 0) of the first bit shifted.

Examples of supported data alignment and the required First Bit Shifted configuration are illustrated in Figure 48-3 for LSB First configurations and Figure 48-4 for MSB First configurations.

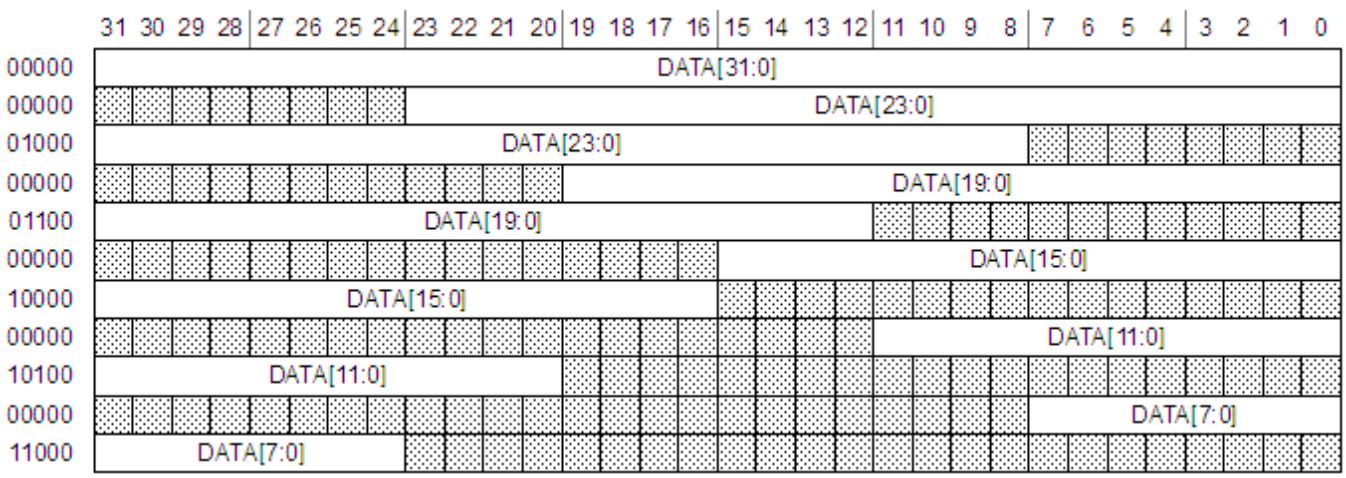


Figure 48-3. SAI first bit shifted, LSB first

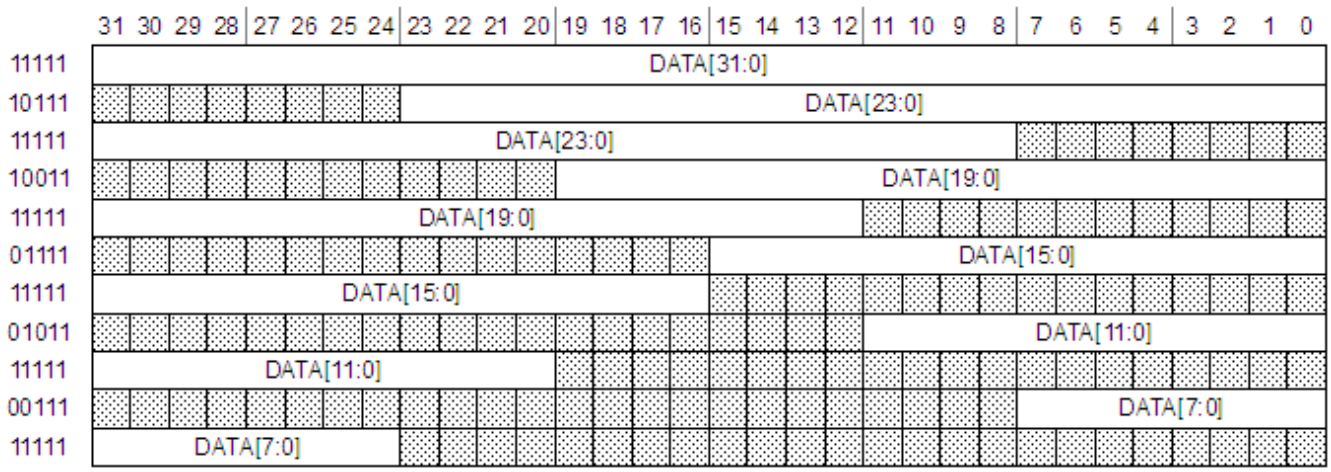


Figure 48-4. SAI first bit shifted, MSB first

48.5.5.2 FIFO pointers

When writing to a TDR, the WFP of the corresponding TFR increments after each valid write. The SAI supports 8-bit, 16-bit and 32-bit writes to the TDR and the FIFO pointer will increment after each individual write. Note that 8-bit writes should only be used when transmitting up to 8-bit data and 16-bit writes should only be used when transmitting up to 16-bit data.

Writes to a TDR are ignored if the corresponding bit of TCR3[TCE] is clear or if the FIFO is full. If the Transmit FIFO is empty, the TDR must be written at least three bit clocks before the start of the next unmasked word to avoid a FIFO underrun.

When reading an RDR, the RFP of the corresponding RFR increments after each valid read. The SAI supports 8-bit, 16-bit and 32-bit reads from the RDR and the FIFO pointer will increment after each individual read. Note that 8-bit reads should only be used when receiving up to 8-bit data and 16-bit reads should only be used when receiving up to 16-bit data.

Reads from an RDR are ignored if the corresponding bit of RCR3[RCE] is clear or if the FIFO is empty. If the Receive FIFO is full, the RDR must be read at least three bit clocks before the end of an unmasked word to avoid a FIFO overrun.

48.5.5.3 FIFO packing

FIFO packing supports storing multiple 8-bit or 16-bit data words in one 32-bit FIFO word for the transmitter and/or receiver. While this can be emulated by adjusting the number of bits per word and number of words per frame (for example, one 32-bit word

per frame versus two 16-bit words per frame), FIFO packing does not require even multiples of words per frame and fully supports word masking. When FIFO packing is enabled, the FIFO pointers only increment when the full 32-bit FIFO word has been written (transmit) or read (receive) by software, supporting scenarios where different words within each frame are loaded/stored in different areas of memory.

When 16-bit FIFO packing is enabled for transmit, the transmit shift register is loaded at the start of each frame and after every second unmasked transmit word. The first word transmitted is taken from 16-bit word at byte offset $0 (first bit is selected by TCFG5[FBT] must be configured within this 16-bit word) and the second word transmitted is taken from the 16-bit word at byte offset $2 (first bit is selected by TCSR5[FBT][3:0]). The transmitter will transmit logic zero until the start of the next word once the 16-bit word has been transmitted.

When 16-bit FIFO packing is enabled for receive, the receive shift register is stored after every second unmasked received word, and at the end of each frame if there is an odd number of unmasked received words in each frame. The first word received is stored in the 16-bit word at byte offset $0 (first bit is selected by RCFG5[FBT] and must be configured within this 16-bit word) and the second word received is stored in the 16-bit word at byte offset $2 (first bit is selected by RCSR5[FBT][3:0]). The receiver will ignore received data until the start of the next word once the 16-bit word has been received.

The 8-bit FIFO packing is similar to 16-bit packing except four words are loaded or stored into each 32-bit FIFO word. The first word is loaded/stored in byte offset $0, second word in byte offset $1, third word in byte offset $2 and fourth word in byte offset $3. The TCFG5[FBT] and/or RCFG5[FBT] must be configured within byte offset $0.

48.5.5.4 FIFO Combine

FIFO combining mode allows the separate FIFOs for multiple data channels to be used as a single FIFO for either software accesses or a single data channel or both. Note that the enabled data channels must be contiguous and data channel 0 must be enabled when FIFO Combine mode is enabled.

Combining FIFOs for software access (writing transmit FIFO registers, reading receive FIFO registers) allows a DMA controller or software to read or write multiple FIFOs without incrementing the address that is accessed. Once enabled, the first software access to a FIFO register will access the first enabled channel FIFO, while the second access to a FIFO register will access the second enabled channel FIFO. This continues until software

accesses the last enabled channel FIFO and the pointer resets back to the first enabled channel FIFO. To reset the pointer manually, software can reset the FIFOs or disable the FIFO combining on software accesses.

Combining FIFOs for transmit data channels allows one data channel to use the FIFOs of all enabled channel FIFOs, with identical data output on each enabled data channel. The transmit shift registers for all enabled data channels are loaded at the start of each frame and every N unmasked words (where N is the number of enabled data channels). The first word transmitted is loaded from the first enabled channel FIFO, while the second word transmitted is loaded from the second enabled channel FIFO, and so on until the end of the frame. Since the first word in each frame is always loaded from the first enabled data channel, it is recommended that the number of unmasked words in each frame is evenly divisible by the number of enabled data channels.

Combining FIFOs for receive data channels allows one data channel to use the FIFOs of all enabled channel FIFOs, with received data from channel 0 stored into each enabled data channel. The receive shift register for all enabled data channels are stored after every N unmasked words (where N is the number of enabled data channels). The first word received is stored to the first enabled channel FIFO, while the second word received is stored to the second enabled channel FIFO, and so on until the end of the frame. Since the first word in each frame is always stored the first enabled data channel, it is recommended that the number of unmasked words in each frame is evenly divisible by the number of enabled data channels.

Note that combining FIFOs for data channels will load or store each channel FIFO at the same time. This means that FIFO error conditions are only checked every N words (where N is the number of enabled data channels) and that the FIFO warning and request flags will assert if any of the enabled data channel meets the warning flag or request flag conditions.

48.5.6 Word mask register

The SAI transmitter and receiver each contain a word mask register, namely TMR and RMR, that can be used to mask any word in the frame. Because the word mask register is double buffered, software can update it before the end of each frame to mask a particular word in the next frame.

The TMR causes the Transmit Data pin to be tri-stated for the length of each selected word and the transmit FIFO is not read for masked words.

The RMR causes the received data for each selected word to be discarded and not written to the receive FIFO.

48.5.7 Interrupts and DMA requests

The SAI transmitter and receiver generate separate interrupts and separate DMA requests, but support the same status flags.

48.5.7.1 FIFO request flag

The FIFO request flag is set based on the number of entries in the FIFO and the FIFO watermark configuration.

The transmit FIFO request flag is set when the number of entries in any of the enabled transmit FIFOs is less than or equal to the transmit FIFO watermark configuration and is cleared when the number of entries in each enabled transmit FIFO is greater than the transmit FIFO watermark configuration.

The receive FIFO request flag is set when the number of entries in any of the enabled receive FIFOs is greater than the receive FIFO watermark configuration and is cleared when the number of entries in each enabled receive FIFO is less than or equal to the receive FIFO watermark configuration.

The FIFO request flag can generate an interrupt or a DMA request.

48.5.7.2 FIFO warning flag

The FIFO warning flag is set based on the number of entries in the FIFO.

The transmit warning flag is set when the number of entries in any of the enabled transmit FIFOs is empty and is cleared when the number of entries in each enabled transmit FIFO is not empty.

The receive warning flag is set when the number of entries in any of the enabled receive FIFOs is full and is cleared when the number of entries in each enabled receive FIFO is not full.

The FIFO warning flag can generate an Interrupt or a DMA request.

48.5.7.3 FIFO error flag

The transmit FIFO error flag is set when the any of the enabled transmit FIFOs underflow. After it is set, all enabled transmit channels will transmit zero until TCSR[FEF] is cleared and the next transmit frame starts. All enabled transmit FIFOs must be reset and initialized with new data before TCSR[FEF] is cleared.

When TCR4[FCONT] is set, the FIFO will continue transmitting data following an underflow without software intervention. To ensure that data is transmitted in the correct order, the transmitter will continue from the same word number in the frame that caused the FIFO to underflow, but only after new data has been written to the transmit FIFO. Software should still clear the TCSR[FEF] flag, but without reinitializing the transmit FIFOs.

RCSR[FEF] is set when the any of the enabled receive FIFOs overflow. After it is set, all enabled receive channels discard received data until RCSR[FEF] is cleared and the next next receive frame starts. All enabled receive FIFOs should be emptied before RCSR[FEF] is cleared.

When RCR4[FCONT] is set, the FIFO will continue receiving data following an overflow without software intervention. To ensure that data is received in the correct order, the receiver will continue from the same word number in the frame that caused the FIFO to overflow, but only after data has been read from the receive FIFO. Software should still clear the RCSR[FEF] flag, but without emptying the receive FIFOs.

The FIFO error flag can generate only an interrupt.

48.5.7.4 Sync error flag

The sync error flag, TCSR[SEF] or RCSR[SEF], is set when configured for an externally generated frame sync and the external frame sync asserts when the transmitter or receiver is busy with the previous frame. The external frame sync assertion is ignored and the sync error flag is set. When the sync error flag is set, the transmitter or receiver continues checking for frame sync assertion when idle or at the end of each frame.

The sync error flag can generate an interrupt only.

48.5.7.5 Word start flag

The word start flag is set at the start of the second bit clock for the selected word, as configured by the Word Flag register field.

The word start flag can generate an interrupt only.

Chapter 49
Periodic Interrupt Timer (PIT-RTI)

49.1 Chip-specific PIT-RTI information

49.1.1 PIT configuration

The device has a single instantiation of the PIT_RTI with 16 x PIT timers/channels, each 32 bits in length, and a 32-bit Real Time Interrupt (RTI) channel that can be clocked from an FXOSC. Consecutive channels can be chained to build 64-bit timers. This is controlled by software.

The reset value for the PIT_MCR register is 0x0000_0006. This value overrides the reset value that is shown in the body of this chapter.

RTI supports wakeup even when the system clock in gated in STOP mode.

49.1.2 Input Output

This section describes the input output requirements for PIT_RTI. The PIT_RTI has no dedicated inputs except the clocks. All the PIT instances are capable of generating periodic interrupts. However, PIT_0 to PIT_15 channels are also routed to generate trigger to specific peripherals as listed in the table below.

Table 49-1. PIT Trigger Allocation

PIT Channel	Peripheral	Comments
PIT_0	DMA_TRIGGER #1	DMA Periodic trigger mode
PIT_1	DMA_TRIGGER#2	DMA Periodic trigger mode
PIT_4	DMA_TRIGGER#3	DMA Periodic trigger mode
PIT_5	DMA_TRIGGER#4	DMA Periodic trigger mode
PIT_3	BCTU[23]	BCTU input trigger number 23
PIT_7	BCTU[55]	BCTU input trigger number 55

Table continues on the next page...

Table 49-1. PIT Trigger Allocation (continued)

PIT Channel	Peripheral	Comments
PIT_2	ADC 10-bit	Injection channels
PIT_6	ADC 12-bit	Injection channels
PIT_9	DSPI_0	Hardware trigger mode
PIT_10	DSPI_1	Hardware trigger mode
PIT_11	DSPI_2	Hardware trigger mode
PIT_12	DSPI_3	Hardware trigger mode
PIT_13	FlexRay	Stopwatch feature
PIT_14	FlexCAN_0, FlexCAN_1, FlexCAN_2, FlexCAN_3	FlexCAN
PIT_15	FlexCAN_4, FlexCAN_5, FlexCAN_6, FlexCAN_7	FlexCAN

49.1.3 PIT-RTI's stop-stop ack protocol

If RTI clock is present the stop request acknowledgement is generated "2 FXOSC clock Cycles" after the request, whereas if the clock is absent the acknowledgement is asserted immediately.

49.2 Introduction

The PIT module is an array of timers that can be used to raise interrupts and triggers. It includes a dedicated Real Time Interrupt Timer (RTI), which runs on a separate clock and can be used for system wakeup in STOP mode.

49.2.1 Block diagram

The following figure shows the block diagram of the PIT-RTI module.

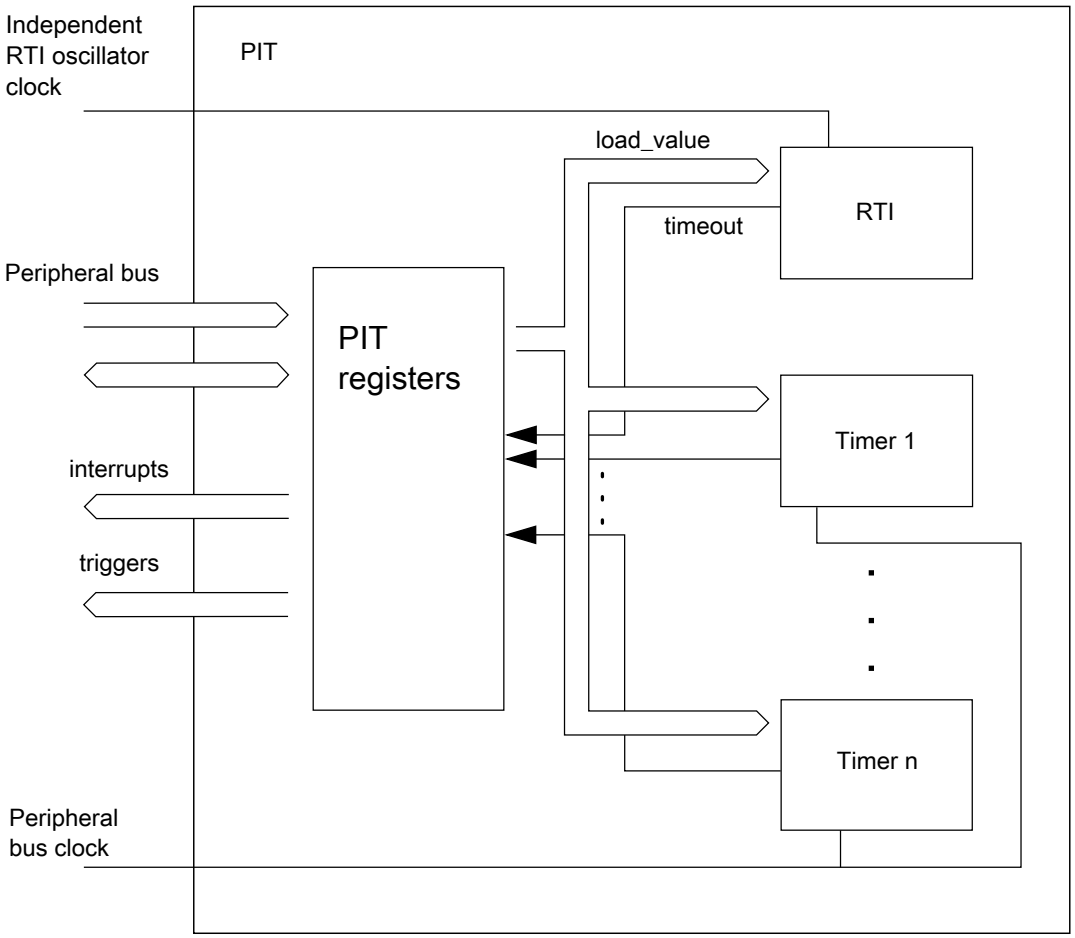


Figure 49-1. Block diagram of the PIT-RTI

NOTE

See the chip-specific PIT information for the number of PIT channels used in this MCU.

49.2.2 Features

The main features of this block are:

- One RTI (Real-Time Interrupt) timer to wakeup the CPU in stop mode
- Ability of timers to generate trigger pulses
- Ability of timers to generate interrupts
- Maskable interrupts
- Option to raise RTI interrupt, even when the bus clock is switched off

- Power saving with a separate input clock for the RTI timer. All other timers share one common core clock.
- Independent timeout periods for each timer

49.3 Signal description

The PIT module has no external pins.

49.4 Memory map/register description

This section provides a detailed description of all registers accessible in the PIT_RTI module.

- Reserved registers will read as 0, writes will have no effect.
- See the chip-specific PIT information for the number of PIT channels used in this MCU.
- The RTI registers should be programmed only when the RTI clock is running.

PIT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	PIT Module Control Register (PIT_MCR)	32	R/W	0000_0006h	49.4.1/2133
E0	PIT Upper Lifetime Timer Register (PIT_LTMR64H)	32	R	0000_0000h	49.4.2/2134
E4	PIT Lower Lifetime Timer Register (PIT_LTMR64L)	32	R	0000_0000h	49.4.3/2134
F0	Timer Load Value Register (PIT_RTI_LDVAL)	32	R/W	0000_0000h	49.4.4/2135
F4	Current Timer Value Register (PIT_RTI_CVAL)	32	R	0000_0000h	49.4.5/2135
F8	Timer Control Register (PIT_RTI_TCTRL)	32	R/W	0000_0000h	49.4.6/2136
FC	Timer Flag Register (PIT_RTI_TFLG)	32	R/W	0000_0000h	49.4.7/2137
100	Timer Load Value Register (PIT_LDVAL0)	32	R/W	0000_0000h	49.4.8/2137
104	Current Timer Value Register (PIT_CVAL0)	32	R	0000_0000h	49.4.9/2138
108	Timer Control Register (PIT_TCTRL0)	32	R/W	0000_0000h	49.4.10/ 2138
10C	Timer Flag Register (PIT_TFLG0)	32	R/W	0000_0000h	49.4.11/ 2139
110	Timer Load Value Register (PIT_LDVAL1)	32	R/W	0000_0000h	49.4.8/2137
114	Current Timer Value Register (PIT_CVAL1)	32	R	0000_0000h	49.4.9/2138
118	Timer Control Register (PIT_TCTRL1)	32	R/W	0000_0000h	49.4.10/ 2138

Table continues on the next page...

PIT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11C	Timer Flag Register (PIT_TFLG1)	32	R/W	0000_0000h	49.4.11/ 2139
120	Timer Load Value Register (PIT_LDVAL2)	32	R/W	0000_0000h	49.4.8/2137
124	Current Timer Value Register (PIT_CVAL2)	32	R	0000_0000h	49.4.9/2138
128	Timer Control Register (PIT_TCTRL2)	32	R/W	0000_0000h	49.4.10/ 2138
12C	Timer Flag Register (PIT_TFLG2)	32	R/W	0000_0000h	49.4.11/ 2139
130	Timer Load Value Register (PIT_LDVAL3)	32	R/W	0000_0000h	49.4.8/2137
134	Current Timer Value Register (PIT_CVAL3)	32	R	0000_0000h	49.4.9/2138
138	Timer Control Register (PIT_TCTRL3)	32	R/W	0000_0000h	49.4.10/ 2138
13C	Timer Flag Register (PIT_TFLG3)	32	R/W	0000_0000h	49.4.11/ 2139
140	Timer Load Value Register (PIT_LDVAL4)	32	R/W	0000_0000h	49.4.8/2137
144	Current Timer Value Register (PIT_CVAL4)	32	R	0000_0000h	49.4.9/2138
148	Timer Control Register (PIT_TCTRL4)	32	R/W	0000_0000h	49.4.10/ 2138
14C	Timer Flag Register (PIT_TFLG4)	32	R/W	0000_0000h	49.4.11/ 2139
150	Timer Load Value Register (PIT_LDVAL5)	32	R/W	0000_0000h	49.4.8/2137
154	Current Timer Value Register (PIT_CVAL5)	32	R	0000_0000h	49.4.9/2138
158	Timer Control Register (PIT_TCTRL5)	32	R/W	0000_0000h	49.4.10/ 2138
15C	Timer Flag Register (PIT_TFLG5)	32	R/W	0000_0000h	49.4.11/ 2139
160	Timer Load Value Register (PIT_LDVAL6)	32	R/W	0000_0000h	49.4.8/2137
164	Current Timer Value Register (PIT_CVAL6)	32	R	0000_0000h	49.4.9/2138
168	Timer Control Register (PIT_TCTRL6)	32	R/W	0000_0000h	49.4.10/ 2138
16C	Timer Flag Register (PIT_TFLG6)	32	R/W	0000_0000h	49.4.11/ 2139
170	Timer Load Value Register (PIT_LDVAL7)	32	R/W	0000_0000h	49.4.8/2137
174	Current Timer Value Register (PIT_CVAL7)	32	R	0000_0000h	49.4.9/2138
178	Timer Control Register (PIT_TCTRL7)	32	R/W	0000_0000h	49.4.10/ 2138
17C	Timer Flag Register (PIT_TFLG7)	32	R/W	0000_0000h	49.4.11/ 2139
180	Timer Load Value Register (PIT_LDVAL8)	32	R/W	0000_0000h	49.4.8/2137
184	Current Timer Value Register (PIT_CVAL8)	32	R	0000_0000h	49.4.9/2138
188	Timer Control Register (PIT_TCTRL8)	32	R/W	0000_0000h	49.4.10/ 2138

Table continues on the next page...

PIT memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
18C	Timer Flag Register (PIT_TFLG8)	32	R/W	0000_0000h	49.4.11/ 2139
190	Timer Load Value Register (PIT_LDVAL9)	32	R/W	0000_0000h	49.4.8/2137
194	Current Timer Value Register (PIT_CVAL9)	32	R	0000_0000h	49.4.9/2138
198	Timer Control Register (PIT_TCTRL9)	32	R/W	0000_0000h	49.4.10/ 2138
19C	Timer Flag Register (PIT_TFLG9)	32	R/W	0000_0000h	49.4.11/ 2139
1A0	Timer Load Value Register (PIT_LDVAL10)	32	R/W	0000_0000h	49.4.8/2137
1A4	Current Timer Value Register (PIT_CVAL10)	32	R	0000_0000h	49.4.9/2138
1A8	Timer Control Register (PIT_TCTRL10)	32	R/W	0000_0000h	49.4.10/ 2138
1AC	Timer Flag Register (PIT_TFLG10)	32	R/W	0000_0000h	49.4.11/ 2139
1B0	Timer Load Value Register (PIT_LDVAL11)	32	R/W	0000_0000h	49.4.8/2137
1B4	Current Timer Value Register (PIT_CVAL11)	32	R	0000_0000h	49.4.9/2138
1B8	Timer Control Register (PIT_TCTRL11)	32	R/W	0000_0000h	49.4.10/ 2138
1BC	Timer Flag Register (PIT_TFLG11)	32	R/W	0000_0000h	49.4.11/ 2139
1C0	Timer Load Value Register (PIT_LDVAL12)	32	R/W	0000_0000h	49.4.8/2137
1C4	Current Timer Value Register (PIT_CVAL12)	32	R	0000_0000h	49.4.9/2138
1C8	Timer Control Register (PIT_TCTRL12)	32	R/W	0000_0000h	49.4.10/ 2138
1CC	Timer Flag Register (PIT_TFLG12)	32	R/W	0000_0000h	49.4.11/ 2139
1D0	Timer Load Value Register (PIT_LDVAL13)	32	R/W	0000_0000h	49.4.8/2137
1D4	Current Timer Value Register (PIT_CVAL13)	32	R	0000_0000h	49.4.9/2138
1D8	Timer Control Register (PIT_TCTRL13)	32	R/W	0000_0000h	49.4.10/ 2138
1DC	Timer Flag Register (PIT_TFLG13)	32	R/W	0000_0000h	49.4.11/ 2139
1E0	Timer Load Value Register (PIT_LDVAL14)	32	R/W	0000_0000h	49.4.8/2137
1E4	Current Timer Value Register (PIT_CVAL14)	32	R	0000_0000h	49.4.9/2138
1E8	Timer Control Register (PIT_TCTRL14)	32	R/W	0000_0000h	49.4.10/ 2138
1EC	Timer Flag Register (PIT_TFLG14)	32	R/W	0000_0000h	49.4.11/ 2139
1F0	Timer Load Value Register (PIT_LDVAL15)	32	R/W	0000_0000h	49.4.8/2137
1F4	Current Timer Value Register (PIT_CVAL15)	32	R	0000_0000h	49.4.9/2138
1F8	Timer Control Register (PIT_TCTRL15)	32	R/W	0000_0000h	49.4.10/ 2138

Table continues on the next page...

PIT memory map (continued)

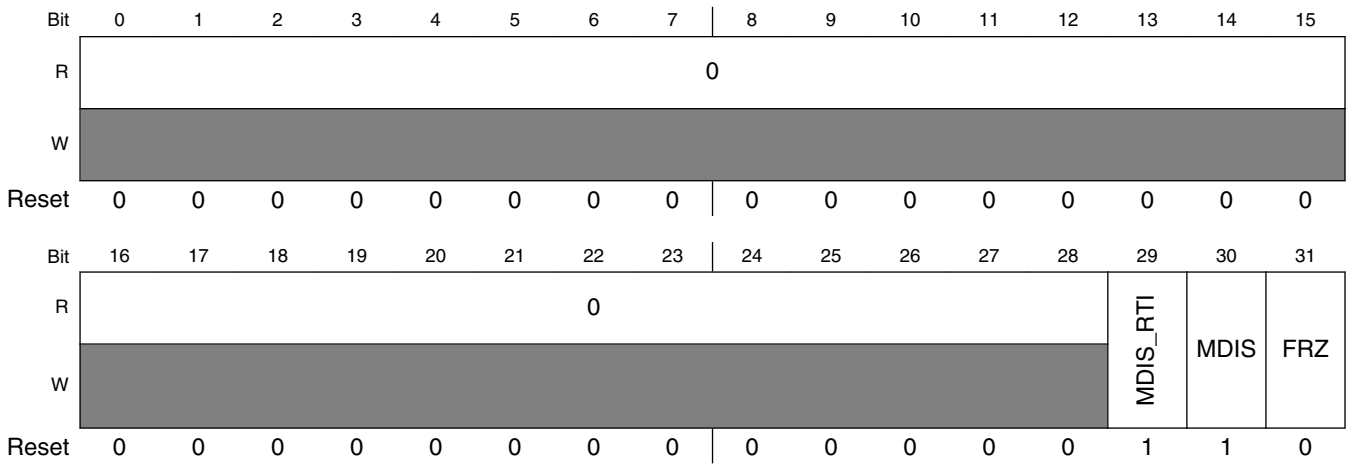
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1FC	Timer Flag Register (PIT_TFLG15)	32	R/W	0000_0000h	49.4.11/ 2139

49.4.1 PIT Module Control Register (PIT_MCR)

This register enables or disables the PIT timer clocks and controls the timers when the PIT enters the Debug mode.

Access: User read/write

Address: 0h base + 0h offset = 0h



PIT_MCR field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 MDIS_RTI	Module Disable - RTI section This is used to disable the RTI timer. This bit must be enabled before any RTI setup is done. 0 Clock for RTI Timers is enabled 1 Clock for RTI Timers is disabled
30 MDIS	Module Disable - (PIT section) Disables the standard timers.The RTI timer is not affected by this field. This field must be enabled before any other setup is done. **NOTE:** Always write to this bit after at least 3 bus clock cycles of enabling the PIT clock gate in the device clock generation module.

Table continues on the next page...

PIT_MCR field descriptions (continued)

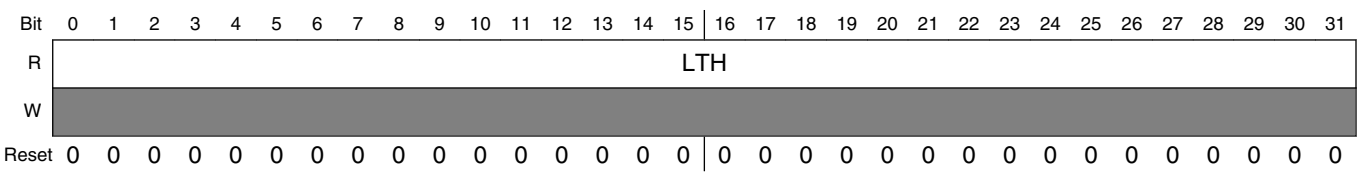
Field	Description
	0 Clock for standard PIT timers is enabled. 1 Clock for standard PIT timers is disabled.
31 FRZ	Freeze Allows the timers to be stopped when the device enters the Debug mode. 0 Timers continue to run in Debug mode. 1 Timers are stopped in Debug mode.

49.4.2 PIT Upper Lifetime Timer Register (PIT_LTMR64H)

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

Access: User read only

Address: 0h base + E0h offset = E0h



PIT_LTMR64H field descriptions

Field	Description
0–31 LTH	Life Timer value Shows the timer value of timer 1. If this register is read at a time t1, LTMR64L shows the value of timer 0 at time t1.

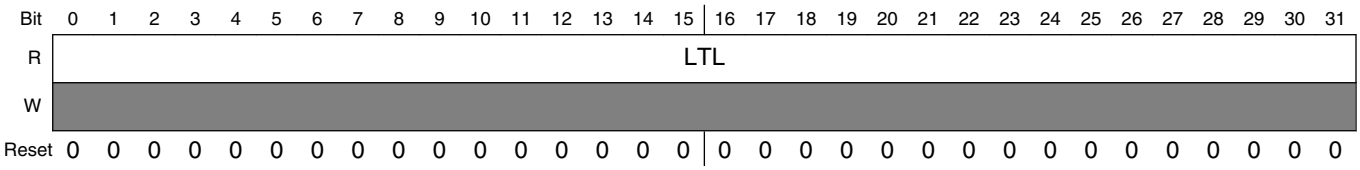
49.4.3 PIT Lower Lifetime Timer Register (PIT_LTMR64L)

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

To use LTMR64H and LTMR64L, timer 0 and timer 1 need to be chained. To obtain the correct value, first read LTMR64H and then LTMR64L. LTMR64H will have the value of CVAL1 at the time of the first access, LTMR64L will have the value of CVAL0 at the time of the first access, therefore the application does not need to worry about carry-over effects of the running counter.

Access: User read only

Address: 0h base + E4h offset = E4h



PIT_LTMR64L field descriptions

Field	Description
0–31 LTL	Life Timer value Shows the value of timer 0 at the time LTMR64H was last read. It will only update if LTMR64H is read.

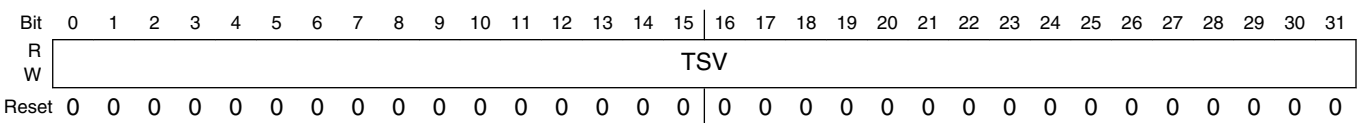
49.4.4 Timer Load Value Register (PIT_RTI_LDVAL)

These registers select the timeout period for the timer interrupts.

In the case of the RTI, it will take several cycles until this value is synchronized into the RTI clock domain. For all other timers the value change is visible immediately. The synchronisation mechanism allows 0 wait states in this case.

Access: User read/write

Address: 0h base + F0h offset = F0h



PIT_RTI_LDVAL field descriptions

Field	Description
0–31 TSV	Timer Start Value Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer; instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again. **NOTE:** The RTI timer must not be set to a value lower than 32 cycles, otherwise interrupts may be lost, as it takes several cycles to clear the RTI interrupt. For the other timers, this limit does not apply, however there will be practical limits, since the processor will require several cycles to service an interrupt.

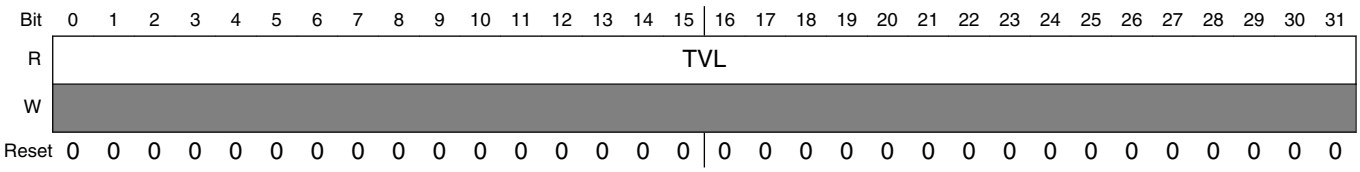
49.4.5 Current Timer Value Register (PIT_RTI_CVAL)

These registers indicate the current timer position.

For the RTI timers, they show a value which is several cycles old, since it originates from a potentially different clock domain.

Access: User read only

Address: 0h base + F4h offset = F4h



PIT_RTI_CVAL field descriptions

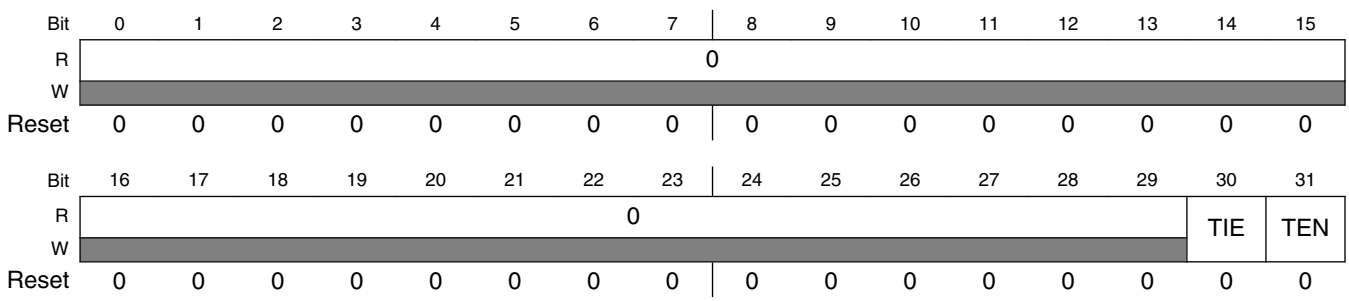
Field	Description
0–31 TVL	Current Timer Value Represents the current timer value, if the timer is enabled. **NOTE:** • If the timer is disabled, do not use this field as its value is unreliable. • The timer uses a downcounter. The timer values are frozen in Debug mode if MCR[FRZ] is set.

49.4.6 Timer Control Register (PIT_RTI_TCTRL)

These registers contain the control bits for each timer.

Access: User read/write

Address: 0h base + F8h offset = F8h



PIT_RTI_TCTRL field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 TIE	Timer Interrupt Enable When an interrupt is pending or, RTI_TFLG[TIF] is set, enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated RTI_TFLG[TIF] flag must be cleared first.

Table continues on the next page...

PIT_RTI_TCTRL field descriptions (continued)

Field	Description
	0 Interrupt requests from the RTI Timer are disabled. 1 Interrupt will be requested whenever TIF is set.
31 TEN	Timer Enable Bit This bit enables or disables the timer. 0 RTI Timer is disabled. 1 RTI Timer is active.

49.4.7 Timer Flag Register (PIT_RTI_TFLG)

These registers hold the PIT interrupt flags.

Access: User read/write

Address: 0h base + FCh offset = FCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_RTI_TFLG field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Timer Interrupt Flag. TIF is set to 1 at the end of the timer period. This flag can be cleared only by writing it with 1 . Writing 0 has no effect. If enabled (TIE = 1), TIF causes an interrupt request. 0 Time-out has not yet occurred. 1 Time-out has occurred.

49.4.8 Timer Load Value Register (PIT_LDVAL*n*)

These registers select the timeout period for the timer interrupts.

Access: User read/write

Address: 0h base + 100h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TSV																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_LDVAL*n* field descriptions

Field	Description
0–31 TSV	Timer Start Value Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer; instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

49.4.9 Current Timer Value Register (PIT_CVAL*n*)

These registers indicate the current timer position.

Access: User read only

Address: 0h base + 104h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TVL																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_CVAL*n* field descriptions

Field	Description
0–31 TVL	Current Timer Value Represents the current timer value, if the timer is enabled. **NOTE:** • If the timer is disabled, do not use this field as its value is unreliable. • The timer uses a downcounter. The timer values are frozen in Debug mode if MCR[FRZ] is set.

49.4.10 Timer Control Register (PIT_TCTRL*n*)

These registers contain the control bits for each timer.

Access: User read/write

Address: 0h base + 108h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TCTRL*n* field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 CHN	Chain Mode When activated, Timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be chained. The RTI timer cannot be chained. 0 Timer is not chained. 1 Timer is chained to previous timer. For example, for Channel 2, if this field is set, Timer 2 is chained to Timer 1.
30 TIE	Timer Interrupt Enable When an interrupt is pending, or, TFLGn[TIF] is set, enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TFLGn[TIF] must be cleared first. 0 Interrupt requests from Timer n are disabled. 1 Interrupt will be requested whenever TIF is set.
31 TEN	Timer Enable Enables or disables the timer. 0 Timer n is disabled. 1 Timer n is enabled.

49.4.11 Timer Flag Register (PIT_TFLG*n*)

These registers hold the PIT interrupt flags.

Access: User read/write

Address: 0h base + 10Ch offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PIT_TFLG*n* field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Timer Interrupt Flag Sets to 1 at the end of the timer period. Writing 1 to this flag clears it. Writing 0 has no effect. If enabled, or, when TCTRLn[TIE] = 1, TIF causes an interrupt request. 0 Timeout has not yet occurred. 1 Timeout has occurred.

49.5 Functional description

This section provides the functional description of the module.

49.5.1 General operation

This section gives detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses and interrupts. Each interrupt is available on a separate interrupt line. Additionally the RTI timer can be used to wake up the processor.

49.5.1.1 Timers / RTI

The timers generate triggers at periodic intervals, when enabled. The timers load the start values as specified in their LDVAL registers, count down to 0 and then load the respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked by setting TCTRLn[TIE]. A new interrupt can be generated only after the previous one is cleared. Since in the case of the RTI, clearing the interrupt crosses clock domains, a minimum load value of 32 must be maintained.

If desired, the current counter value of the timer can be read via the CVAL registers. The value of the RTI counter can be delayed considerably, as it is synchronized to the bus clock from the RTI clock domain.

The counter period can be restarted, by first disabling, and then enabling the timer with TCTRLn[TEN]. See the following figure.

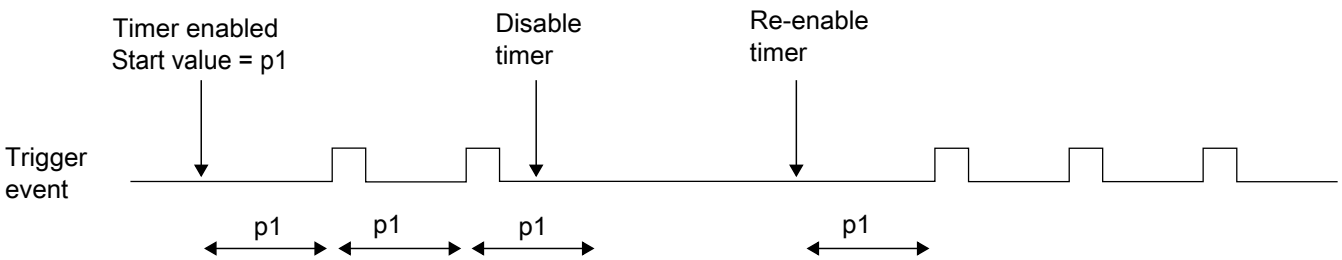


Figure 49-2. Stopping and starting a timer

The counter period of a running timer can be modified, by first disabling the timer, setting a new load value, and then enabling the timer again. See the following figure.

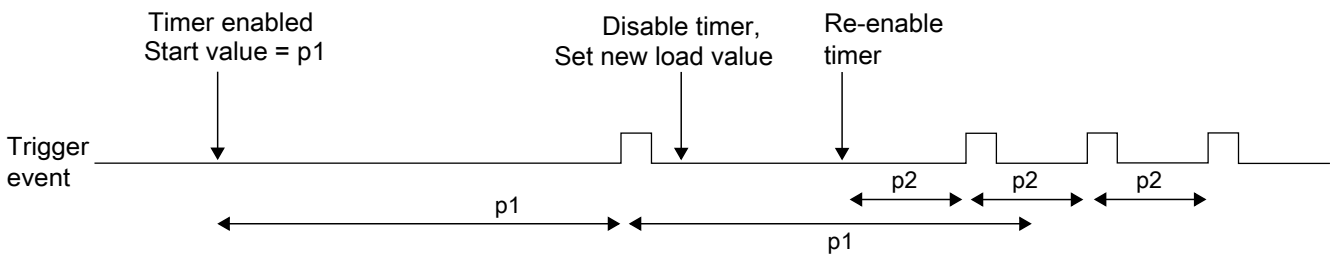


Figure 49-3. Modifying running timer period

It is also possible to change the counter period without restarting the timer by writing LDVAL with the new load value. This value will then be loaded after the next trigger event. See the following figure.

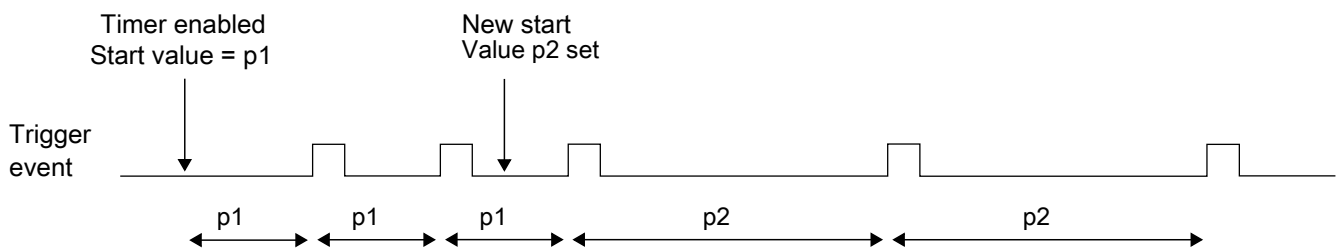


Figure 49-4. Dynamically setting a new load value

49.5.1.2 Debug mode

In Debug mode, the timers will be frozen based on MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system, for example, the timer values, and then continue the operation.

49.5.2 Interrupts

All the timers support interrupt generation. The RTI is typically used for system wakeup, but can be used for interrupt generation as well. See the MCU specification for related vector addresses and priorities.

Timer interrupts can be enabled by setting TCTRLn[TIE]. TFLGn[TIF] are set to 1 when a timeout occurs on the associated timer, and are cleared to 0 by writing a 1 to the corresponding TFLGn[TIF].

The PIT_RTI generates a real time interrupt when the selected interrupt time period elapses. The RTI interrupt is disabled locally by setting TCTCRLn[TIE] to zero. The real time interrupt flag (TIF) is set to 1 when a timeout occurs, and is cleared by writing a 1 to the corresponding TFLGn[TIF]. The flag will be set regardless whether the interrupt is enabled.

The RTI can be used for periodic wakeup from STOP mode. It can also be used to generate a general-purpose interrupt.

49.5.3 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer n-1 has counted down to 0, counter n will decrement the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

49.6 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.
- The RTI clock has a frequency of 10 MHz.

- The RTI creates a wakeup interrupt every 500 ms.
- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 ms.

The PIT module must be activated by writing a 0 to MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns and the 10 MHz frequency equates to a clock period of 100 ns. Therefore, the RTI timer needs to trigger every 500 ms/100 ns = 5,000,000 cycles. Timer 1 needs to trigger every 5.12 ms/20 ns = 256,000 cycles and Timer 3 every 30 ms/20 ns = 1,500,000 cycles. The value for the LDVAL register trigger is calculated as:

LDVAL trigger = (period / clock period) -1

This means RTI LDVAL will be written with 004C4B3F hex; LDVAL1 and LDVAL3 must be written with 0x0003E7FF and 0x0016E35F respectively.

To generate the wakeup interrupt, the interrupt line must be enabled by writing 1 to the RTI TIE bit in the TCTRL register. To start the RTI, the TEN in the RTI TCTRL register must also be set.

The interrupt for Timer 1 is enabled by setting TCTRL1[TIE]. The timer is started by writing 1 to TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore, Timer 3 is started by writing a 1 to TCTRL3[TEN]. TCTRL3[TIE] stays at 0.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// RTI
PIT_RTI_LDVAL = 0x004C4B3F; // setup RTI for 5000000 cycles
PIT_RTI_TCTRL = PIT_TIE; // let RTI generate interrupts
PIT_RTI_TCTRL |= PIT_TEN; // start RTI


// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1


// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3 for 1500000 cycles
PIT_TCTRL3 |= TEN; // start Timer 3
```

49.7 Example configuration for chained timers

In the example configuration:

- The PIT clock has a frequency of 100 MHz.
- Timers 1 and 2 are available.
- An interrupt shall be raised every 1 minute.

The PIT module needs to be activated by writing a 0 to MCR[MDIS].

The 100 MHz clock frequency equates to a clock period of 10 ns, so the PIT needs to count for 6000 million cycles, which is more than a single timer can do. So, Timer 1 is set up to trigger every 6 s (600 million cycles). Timer 2 is chained to Timer 1 and programmed to trigger 10 times.

The value for the LDVAL register trigger is calculated as number of cycles-1, so LDVAL1 receives the value 0x23C345FF and LDVAL2 receives the value 0x00000009.

The interrupt for Timer 2 is enabled by setting TCTRL2[TIE], the Chain mode is activated by setting TCTRL2[CHN], and the timer is started by writing a 1 to TCTRL2[TEN]. TCTRL1[TEN] needs to be set, and TCTRL1[CHN] and TCTRL1[TIE] are cleared.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 2
PIT_LDVAL2 = 0x00000009; // setup Timer 2 for 10 counts
PIT_TCTRL2 = TIE; // enable Timer 2 interrupt
PIT_TCTRL2 |= CHN; // chain Timer 2 to Timer 1
PIT_TCTRL2 |= TEN; // start Timer 2


// Timer 1
PIT_LDVAL1 = 0x23C345FF; // setup Timer 1 for 600 000 000 cycles
PIT_TCTRL1 = TEN; // start Timer 1
```

49.8 Example configuration for the lifetime timer

To configure the lifetimer timer, channels 0 and 1 need to be chained together.

First the PIT module needs to be activated by writing a 0 to the MDIS bit in the CTRL register, then the LDVAL registers need to be set to the maximum value.

The timer is a downcounter.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 1
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
PIT_TCTRL1 = 0x0; // disable timer 1 interrupts
PIT_TCTRL1 |= CHN; // chain timer 1 to timer 0
PIT_TCTRL1 |= TEN; // start timer 1


// Timer 0
PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0 for maximum counting period
PIT_TCTRL0 = TEN; // start timer 0
```

To access the lifetime, read first LTMR64H and then LTMR64L.

```
current_uptime = PIT_LTMR64H<<32;
current_uptime = current_uptime + PIT_LTMR64L;
```

Chapter 50
Software Watchdog Timer (SWT)

50.1 Chip-specific SWT information

50.1.1 SWT configuration

The device has three SWTs as shown in the following table.

Table 50-1. SWT Configuration Instances

Instance	Description
SWT0	Intended for Core0.
SWT1	Intended for Core1.
HSM_SWT	Dedicated for HSM.

Each SWT is clocked by the SIRC. SWT_0, and SWT_1 could be serviced by any core, however, HSM_SWT can be serviced only by HSM.

The default configuration for the SWT_0 is enabled out of power on reset. The default configuration for the SWT_1 is disabled out of power on reset. However, it is possible to configure the default position of SWT_0 only via a DCF record.

The default configuration for the HSM_SWT is enabled out of power on reset only if HSM is enabled. For more information on HSM, refer Security Reference Manual.

In device STANDBY and LPU modes, only SWT_0 is available.

NOTE

SWT should be disabled before SIRC is disabled during any mode transition.

50.1.2 SWT register reset value

The register reset values for the SWT registers that are device specific are shown below. This information overrides the reset values given in SWT Memory map section. All other registers reset values are shown in the SWT Memory map section.

SWT_0 Time-out register (SWT_0_TO) resets to 0x0000_0A00h.

SWT_1 Time-out register (SWT_1_TO) resets to 0x0000_0A00h.

SWT_HSM Time-out register (SWT_HSM_TO) resets to 0x0000_1000h.

SWT_0 Control register (SWT_0_CR) resets to FF00_011Bh. However, the value of SWT_0_CR[WEN] depends on the DCF record bit SWT_0_WDG_DIS of the UTEST_MISC register.

SWT_1 Control register (SWT_1_CR) resets to FF00_010Ah.

SWT_HSM Control register (SWT_HSM_CR) resets to 1000_004Fh.

50.1.3 Service mode watchpoint input

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

To use SWT address execution modes for this purpose, the e200z4/z2 core IAC8 register signal jd_watchpt[15] is connected to the SWT jd_watchpt input. (See Watchpoint Support section for details.) The jd_watchpt[15] signal tells the SWT that an IAC8 compare occurred.

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

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

50.2 Introduction

This section provides an overview, list of features, and modes of operation for the SWT.

50.2.1 Overview

The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified time-out period. If this servicing action does not occur before the timer expires, the SWT generates an interrupt or hardware reset request. The SWT can be configured to generate a reset request or interrupt on an initial time-out. The SWT always generates a reset request on a second consecutive time-out.

50.2.2 Features

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial time-out
- Programmable selection of the servicing mode
- Master access protection
- Hard and soft configuration lock bits
- Reset configuration inputs allow timer to be enabled out of reset

50.2.3 Modes of operation

The SWT supports three device modes of operation: normal, debug and stop. When the SWT is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the SWT_CR. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run. In stop mode, operation of the counter is controlled by the STP bit in the SWT_CR. If the STP bit is set, the counter is stopped in stop mode, otherwise it continues to run.

50.3 External signal description

The SWT module does not have any external interface signals.

50.4 Memory Map and Registers

The SWT programming model has seven 32-bit registers. The programming model can be accessed using only 32-bit (word) accesses.

Any of the following attempted accesses are invalid:

- References using non-32-bit access sizes
- Writes to read-only registers
- Writes of incorrect values to SWT_SR when the SWT is enabled
- Accesses to reserved addresses
- Accesses by masters without permission

For an enabled SWT, an invalid access generates a reset request if SWT_CR[RIA] is 1. An invalid access generates a bus error if SWT_CR[RIA] is 0.

If either HLK or SLK in SWT_CR is 1, then the SWT_CR, SWT_TO, SWT_WN, and SWT_SK registers are read-only.

SWT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SWT Control Register (SWT_CR)	32	R/W	See section	50.4.1/2151
4	SWT Interrupt Register (SWT_IR)	32	R/W	0000_0000h	50.4.2/2154
8	SWT Time-out Register (SWT_TO)	32	R/W	See section	50.4.3/2155
C	SWT Window Register (SWT_WN)	32	R/W	0000_0000h	50.4.4/2155
10	SWT Service Register (SWT_SR)	32	W	0000_0000h	50.4.5/2156
14	SWT Counter Output Register (SWT_CO)	32	R	0000_0000h	50.4.6/2156
18	SWT Service Key Register (SWT_SK)	32	R/W	0000_0000h	50.4.7/2157

50.4.1 SWT Control Register (SWT_CR)

NOTE

The reset value of this register is implementation specific. See the chip-specific SWT information.

The SWT_CR contains fields for configuring and controlling the SWT.

This register is read-only if either SWT_CR[HLK] or SWT_CR[SLK] is 1.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MAP0	MAP1	MAP2	MAP3	MAP4	MAP5	MAP6	MAP7	0							
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0					SMD		RIA	WND	ITR	HLK	SLK	Reserved	STP	FRZ	WEN
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value of this register is implementation specific. See the chip-specific SWT information.

SWT_CR field descriptions

Field	Description
0 MAP0	Master Access Protection for Master 0. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP0 corresponds to logical IDs 0 and 8. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
1 MAP1	Master Access Protection for Master 1. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP1 corresponds to logical IDs 1 and 9. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
2 MAP2	Master Access Protection for Master 2. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP2 corresponds to logical IDs 2 and 10. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
3 MAP3	Master Access Protection for Master 3. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP3 corresponds to logical IDs 3 and 11. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
4 MAP4	Master Access Protection for Master 4. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP4 corresponds to logical IDs 4 and 12. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
5 MAP5	Master Access Protection for Master 5. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP5 corresponds to logical IDs 5 and 13. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
6 MAP6	Master Access Protection for Master 6. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP6 corresponds to logical IDs 6 and 14. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch. 0 Access for the master is not enabled 1 Access for the master is enabled
7 MAP7	Master Access Protection for Master 7. The platform bus master assignments are device specific. The number of this field corresponds to the logical ID of the bus master. MAP7 corresponds to logical IDs 7 and 15. For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch.

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
	0 Access for the master is not enabled 1 Access for the master is enabled
8–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–22 SMD	Service Mode 00 Fixed Service Sequence: The watchdog is serviced by writing the fixed sequence 0xA602, 0xB480 to SWT_SR. 01 Keyed Service Sequence: The watchdog is serviced by writing two pseudorandom key values to SWT_SR. 10 Fixed Address Execution: The watchdog is serviced by executing code at the address loaded into the designated IAC register, which cannot be updated while the watchdog is enabled. 11 Incremental Address Execution: The watchdog is serviced by executing code at the address loaded into the designated IAC register, which can be updated.
23 RIA	Reset on Invalid Access **NOTE:** For a description of how this chip implements SWT reset requests resulting from SWT invalid accesses, see the chip-specific SWT information. 0 Invalid access to the SWT generates a bus error 1 Invalid access to the SWT causes a reset request if WEN is 1
24 WND	Window Mode 0 Regular mode: service sequence can be done at any time 1 Windowed mode: service sequence is valid only when the down counter is less than the value in the SWT_WN register
25 ITR	Interrupt Then Reset **NOTE:** For a description of how this chip implements SWT reset requests and interrupt requests resulting from SWT time-outs, see the chip-specific SWT information. 0 Generate a reset request on a time-out 1 Generate an interrupt on an initial time-out; generate a reset request on a second consecutive time-out
26 HLK	Hard Lock This bit is cleared only at reset. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if SLK is 0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
27 SLK	Soft Lock This bit is cleared by writing the unlock sequence to the service register. 0 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read/write registers if HLK is 0 1 SWT_CR, SWT_TO, SWT_WN and SWT_SK are read-only registers
28 Reserved	Reserved This field is reserved.

Table continues on the next page...

SWT_CR field descriptions (continued)

Field	Description
29 STP	Stop Mode Control Allows the watchdog timer to be stopped when the device enters stop mode. 0 SWT counter continues to run in stop mode 1 SWT counter is stopped in stop mode
30 FRZ	Debug Mode Control Allows the watchdog timer to be stopped when the device enters debug mode. 0 SWT counter continues to run in debug mode 1 SWT counter is stopped in debug mode
31 WEN	Watchdog Enabled 0 SWT is disabled 1 SWT is enabled

50.4.2 SWT Interrupt Register (SWT_IR)

The SWT_IR contains the time-out interrupt flag.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															TIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_IR field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 TIF	Time-out Interrupt Flag The flag and interrupt are cleared by writing a 1 to this bit. Writing a 0 has no effect. 0 No interrupt request 1 Interrupt request due to an initial time-out

50.4.3 SWT Time-out Register (SWT_TO)

NOTE

The reset value of this register is implementation specific. See the chip-specific SWT information.

The SWT Time-out (SWT_TO) register contains the 32-bit time-out period. This register is read-only if either SWT_CR[HLK] or SWT_CR[SLK] is 1.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WTO																															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value of this register is implementation specific. See the chip-specific SWT information.

SWT_TO field descriptions

Field	Description
0–31 WTO	Watchdog time-out period in clock cycles. An internal 32-bit down counter is loaded with this value or 0x100, whichever is greater, when the service sequence is written or when the SWT is enabled.

50.4.4 SWT Window Register (SWT_WN)

The SWT Window (SWT_WN) register contains the 32-bit window start value. This register is cleared on reset. This register is read-only if either SWT_CR[HLK] or SWT_CR[SLK] is 1.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WST																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

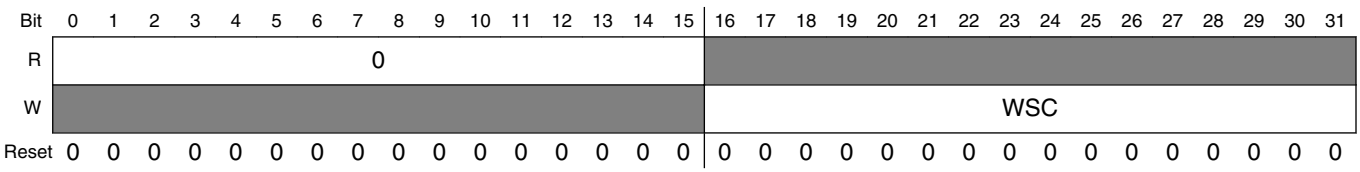
SWT_WN field descriptions

Field	Description
0–31 WST	Window Start Value When window mode is enabled, the service sequence can only be written when the internal down counter is less than this value.

50.4.5 SWT Service Register (SWT_SR)

The SWT Service (SWT_SR) register is the target for service operation writes used to reset the watchdog timer.

Address: 0h base + 10h offset = 10h



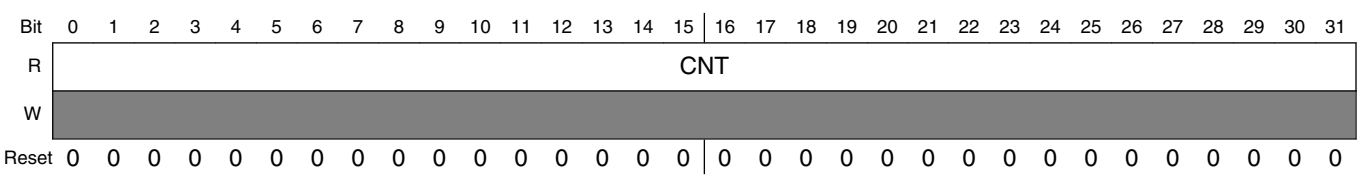
SWT_SR field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 WSC	Watchdog Service Code This field is used to service the watchdog and to clear the Soft Lock bit (SWT_CR[SLK]). If the SWT_CR[SMD] field is 01b, two pseudorandom key values are written to service the watchdog; see Functional description for details. Otherwise, the sequence 0xA602 followed by 0xB480 is written to the WSC field. To clear the Soft Lock bit (SWT_CR[SLK]), the value 0xC520 followed by 0xD928 is written to the WSC field. When read, the WSC field always returns zero.

50.4.6 SWT Counter Output Register (SWT_CO)

The SWT Counter Output (SWT_CO) register is a read-only register that shows the value of the internal down counter when the SWT is disabled.

Address: 0h base + 14h offset = 14h



SWT_CO field descriptions

Field	Description
0–31 CNT	Watchdog Count When the watchdog is disabled (SWT_CR[WEN] is 0), this field shows the value of the internal down counter. When the watchdog is enabled (SWT_CR[WEN] is 1), this field is cleared (the value is

SWT_CO field descriptions (continued)

Field	Description
	0x0000_0000). Values in this field can lag behind the internal counter value for up to 6 system clock cycles plus 8 counter clock cycles. Therefore, the value read from this field immediately after disabling the watchdog may be higher than the actual value of the internal counter.

50.4.7 SWT Service Key Register (SWT_SK)

The SWT Service Key (SWT_SK) register holds the previous (or initial) service key value. This register is read-only if either SWT_CR[HLK] or SWT_CR[SLK] is 1.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																SK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SWT_SK field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 SK	Service Key This field is the previous (or initial) service key value used in keyed service mode. If SWT_CR[SMD] is 01b, the next key value to be written to the SWT_SR is (17*SK+3) mod 2^{16} .

50.5 Functional description

50.5.1 Introduction

The SWT is a 32-bit window watchdog timer designed to enable the system to recover in situations such as software getting trapped in a loop or if a bus transaction fails to terminate. It includes:

- a control register (SWT_CR),
- an interrupt register (SWT_IR),
- a time-out register (SWT_TO),
- a window register (SWT_WN),
- a service register (SWT_SR),

- a counter output register (SWT_CO), and
- a service key register (SWT_SK).

Accesses to SWT registers occur with no peripheral bus wait states. (The peripheral bus bridge may add one or more system wait states.) However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system plus seven counter clock cycles.

50.5.1.1 SWT_CR register

The SWT_CR register includes bits to enable the timer, set configuration options, and lock configuration of the module. The watchdog is enabled by setting the SWT_CR[WEN] bit. The reset value of the SWT_CR[WEN] bit is device specific.[3] If the reset value of this bit is 1, the watchdog starts operation automatically after reset is released. Some devices can be configured to clear this bit automatically during the boot process.

50.5.1.2 SWT_TO register

The SWT_TO register holds the watchdog time-out period in clock cycles unless the value is less than 0x100, in which case the time-out period is set to 0x100. When the SWT is enabled, the time-out period is loaded into an internal 32-bit down counter each time a valid service operation is performed. See the Configuration section to determine the source that is used to clock the down counter.

50.5.1.3 SWT_CO register

The SWT_CO register shows the value of the down counter when the watchdog is disabled. When the watchdog is enabled this register is cleared. The value shown in this register can lag behind the value in the internal counter for up to six system plus eight counter clock cycles.

The SWT_CO register can be used during a software self test of the SWT. For example, the SWT can be enabled and not serviced for a fixed period of time less than the time-out value. Then the SWT can be disabled (SWT_CR[WEN] cleared) and the value of the SWT_CO register read to determine if the internal down counter is working properly.

3. See the chip-specific SWT information.

50.5.2 Configuration locking

The configuration of the SWT can be locked through use of either a soft lock or a hard lock. In either case, when locked, the SWT_CR, SWT_TO, SWT_WN and SWT_SK registers are read-only.

50.5.2.1 Hard lock

The hard lock is enabled by setting the SWT_CR[HLK] bit, which can only be cleared by a reset.

50.5.2.2 Soft lock

The soft lock is enabled by setting the SWT_CR[SLK] bit and is cleared by writing the unlock sequence to the service register.

50.5.3 Unlock sequence

The unlock sequence is a write of 0xC520 followed by a write of 0xD928 to the SWT_SR[WSC] field. There is no timing requirement between the two writes. The unlock sequence logic ignores service sequence writes and recognizes the 0xC520, 0xD928 sequence regardless of previous writes. The unlock sequence can be written at any time and does not require the SWT_CR[WEN] bit to be set.

50.5.4 Servicing operations

When enabled, the SWT requires periodic execution of a servicing operation that is determined by the SWT_CR[SMD] field. Properly servicing the watchdog loads the internal down counter with the time-out period. The servicing modes are:

- fixed service sequence
- keyed service sequence
- fixed execution address
- incremental execution address

50.5.4.1 Fixed service sequence mode

If the SWT_CR[SMD] field is 00b, the fixed service sequence mode is selected, which requires writing 0xA602, then 0xB480 to the SWT_SR[WSC] field to service the watchdog. There is no timing requirement between the two writes and the service sequence logic ignores unlock sequence writes.

50.5.4.2 Keyed service sequence mode

If the SWT_CR[SMD] field is 01b, then the keyed service sequence mode is selected, which requires writing two pseudorandom keys to the SWT_SR[WSC] field to service the watchdog. The key values are determined by the pseudorandom key generator defined by the equation in the following figure. This algorithm will generate a sequence of 2^{16} different key values before repeating. The state of the key generator is held in the SWT_SK register. For example, if SWT_SK[SK] is 0x0100, then the service sequence keys are 0x1103, 0x2136. In this mode, each time a valid key is written to the SWT_SR register, the SWT_SK register is updated. So, after servicing the watchdog by writing 0x1103 and then 0x2136 to the SWT_SR[WSC] field, SWT_SK[SK] is 0x2136 and the next key sequence is 0x3499, 0x7E2C.

$$SK_{n+1} = (17 \times SK_n + 3) \bmod 2^{16}$$

Figure 50-1. Pseudorandom Key Generator

50.5.4.3 Fixed execution address mode

If the SWT_CR[SMD] field is set to 10b, the fixed execution address mode is selected, which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register cannot be updated while the watchdog is enabled.

50.5.4.4 Incremental execution address mode

If the SWT_CR[SMD] field is set to 11b, the incremental execution address mode is selected, which requires executing code at the address loaded into the designated IAC register to service the watchdog. In this mode, the IAC register can be updated while the watchdog is enabled.

50.5.4.5 Window mode

If window mode is enabled (SWT_CR[WND] bit is 1), the service sequence must be applied in the last part of the time-out period defined by SWT_WN.

- The window is open when the down counter is less than the value in SWT_WN.
- Outside this window, service sequence writes are invalid accesses and generate a bus error or reset request depending on the value of SWT_CR[RIA].

For example, if SWT_TO is set to 5000 and SWT_WN is set to 1000, the service sequence must be applied in the last 20% of the time-out period.

Synchronization logic in the watchdog design slightly delays when the window opens. This delay could be up to 3 system clock cycles plus 4 counter clock cycles.

50.5.5 Time-out

The Interrupt Then Reset bit (SWT_CR[ITR]) controls the SWT's action taken when a time-out occurs.

- If the SWT_CR[ITR] bit is 0, the SWT generates a reset request immediately on any time-out.
- If the SWT_CR[ITR] bit is 1, an initial time-out causes the SWT to generate an interrupt and load the down counter with the time-out period.
  - The interrupt is indicated by the Time-out Interrupt Flag (SWT_IR[TIF]).
  - Clear the interrupt by writing 1 to SWT_IR[TIF].

  If the service sequence is not written before the second consecutive time-out, the SWT generates a reset request.

50.5.6 Initialization

All registers should be initialized before setting the SWT_CR[WEN] bit to enable the watchdog. Registers can be initialized in any sequence.

Chapter 51
Real Time Clock (RTC-API)

51.1 Chip-specific RTC-API information

51.1.1 RTC Configuration

The device contains one RTC and one API, where both can perform 32-bit comparisons. Both RTC and API timers can generate interrupts as well as wakeup from low power modes. API can also be used in conjunction with the Comparator module. In STANDBY0/LPU_STANDBY modes configurations, the API is configured to generate a START / NEXT type signal to inform the Comparator module that a 'compare' must be carried out. See Interaction with RTC API to cause wakeup for details.

NOTE

After functional resets, software would be able to read the correct value of the counter after three RTC clocks.

51.1.2 RTC register reset value

All register reset values for the RTC registers that are device specific are shown below. All other registers reset values are shown in the RTC Memory chapter.

RTC Control register resets to 0x0000_ 0000h.

51.1.3 Seamless RTC operation across functional and destructive resets

MPC5746C supports seamless RTC operation across functional and destructive reset with FIRC, SIRC and SXOSC . To support this feature using SXOSC, software should program GPR_CTL[RTC_SXOSC_FREE_RUNNING]. Using SIRC, this feature is limited to SIRC_CTL[SIRCDIV] as divide by 1.

NOTE

SXOSC should be turned ON (by MC_ME clock configurations) before configuring this GPR_CTL[RTC_SXOSC_ FREE_RUNNING] bit.

NOTE

MPC5746C does not support seamless RTC operation across functional and destructive reset using FXOSC.

51.1.4 RTC-API's stop-stop ack protocol

The RTC module provides the acknowledgement as soon as the stop request is made, however the RTC module remains functional.

51.2 Introduction

51.2.1 Overview

The RTC is a free running counter used for time keeping applications. The RTC may be configured to generate an interrupt at a pre-defined interval. The RTC also supports an autonomous periodic interrupt function used to generate an interrupt request.

51.2.2 Features

RTC features include:

- 32 bit counter
- 4 selectable counter clock sources
  - 128 KHz SIRC
  - 32 KHz SXOSC
  - 16 MHz FIRC
  - 8-40 MHz FXOSC
- Optional 512 prescalar and optional 32 prescalar connected in series in the clock path feeding the 32 bit counter.
- 32 bit counter supports times up to greater than 1.5 months with 1 ms resolution.
- 32 bit compare value to support interrupt intervals of 1 second up to greater than 1 hr with 1 second resolution.

- RTC interrupt with interrupt enable.
- Counter runs in all modes of operation.
- RTC status and control register are reset only by POR
- RTC counter is reset when counter is disabled by software and by POR.
- Autonomous periodic interrupt support includes:
  - 32 bit compare value to support wakeup intervals of 1.0 ms to 1 second
  - Wakeup logic has separate enable to support changing compare value while RTC running
  - API interrupt with interrupt enable
  - API compare value can be modified while RTC is running
- Optional interrupt for RTC match, API match, and RTC rollover.

51.2.3 Block Diagram

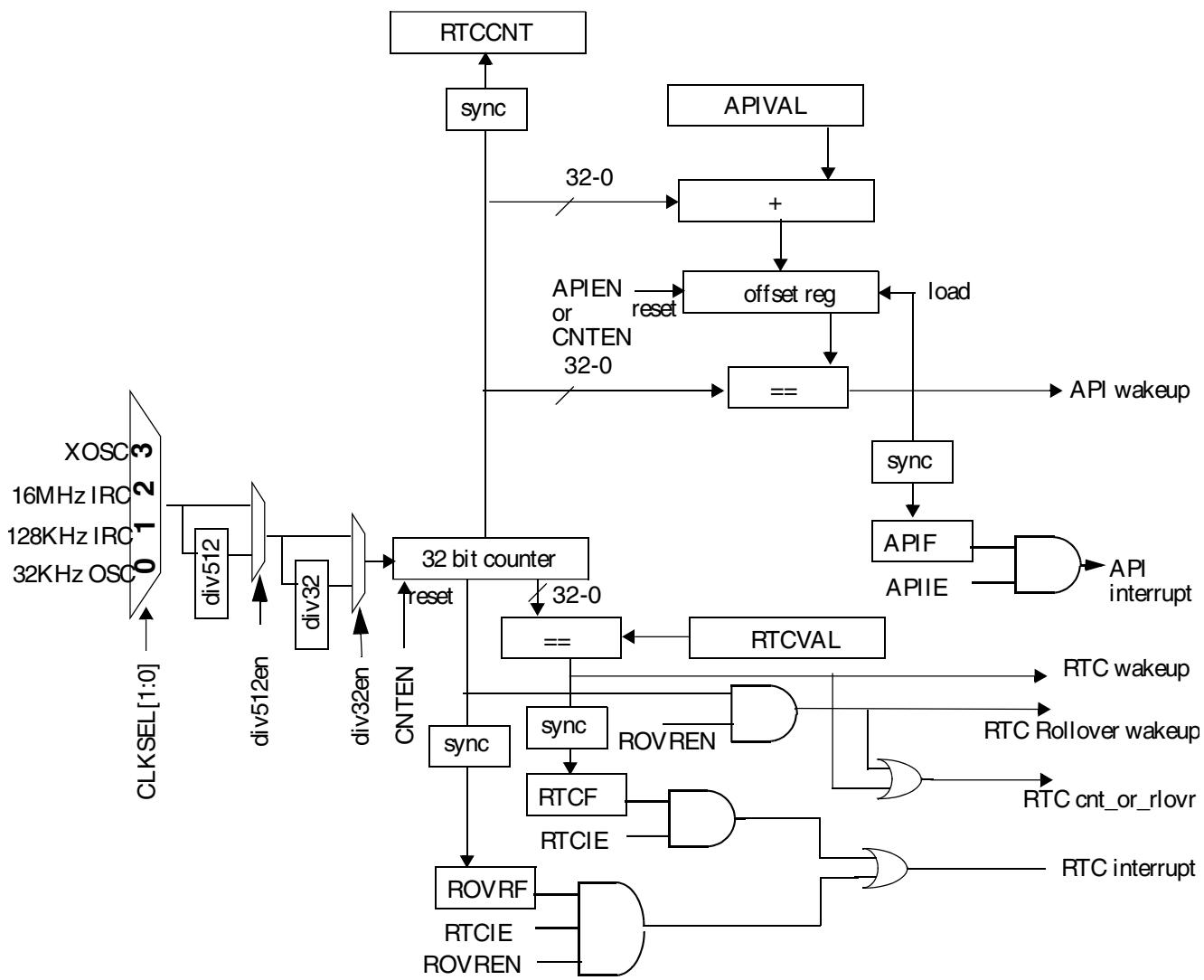


Figure 51-1. Block diagram

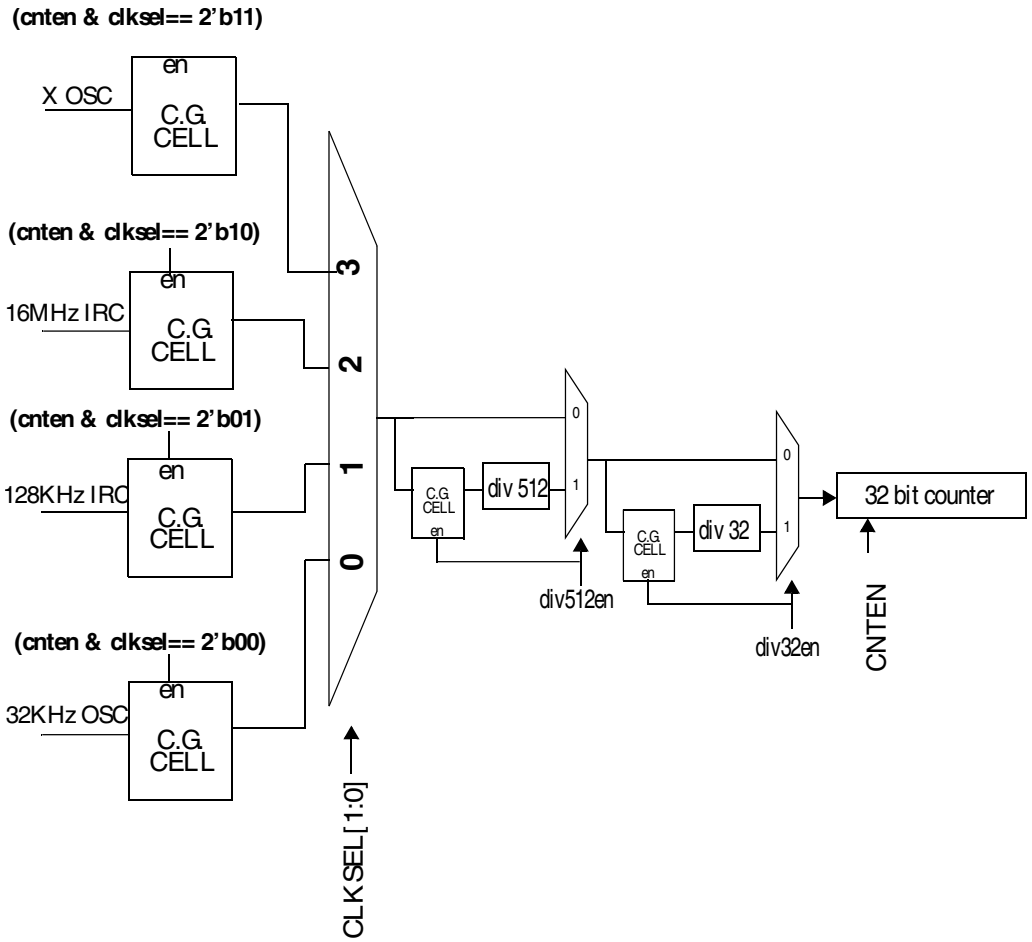


Figure 51-2. Clock gating for RTC clocks

51.2.4 Modes of Operation

51.2.4.1 Functional Mode

All RTC registers can read or written . The RTC/API and associated interrupts are optionally enabled.

51.2.4.2 Debug Mode

On entering into the debug mode the RTC counter freezes on the last valid count if the RTCC[FRZEN] is set. On exit from debug mode counter continues from the frozen value.

51.3 Memory Map and Registers

RTC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	RTC Supervisor Control register (RTC_RTCSUPV)	32	R/W	8000_0000h	51.3.1/2167
4	RTC Control register (RTC_RTCC)	32	R/W	See section	51.3.2/2168
8	RTC Status register (RTC_RTCS)	32	R/W	0000_0000h	51.3.3/2170
C	RTC Counter register (RTC_RTCCNT)	32	R	0000_0000h	51.3.4/2172
10	API Compare Value register (RTC_APIVAL)	32	R/W	0000_0000h	51.3.5/2172
14	RTC Compare Value register (RTC_RTCVAL)	32	R/W	0000_0000h	51.3.6/2173
18	RTC Analogue comparator count register (RTC_ANLCMP_CNT)	32	R/W	0000_0000h	51.3.7/2173

51.3.1 RTC Supervisor Control register (RTC_RTCSUPV)

The RTCSUPV register contains the SUPV bit which determines whether other registers are accessible in supervisor mode or user mode.

NOTE

RTCSUPV register is accessible only in supervisor mode.

Address: 0h base + 0h offset = 0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SUPV	0														
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_RTCSUPV field descriptions

Field	Description
31 SUPV	RTC Supervisor Bit.

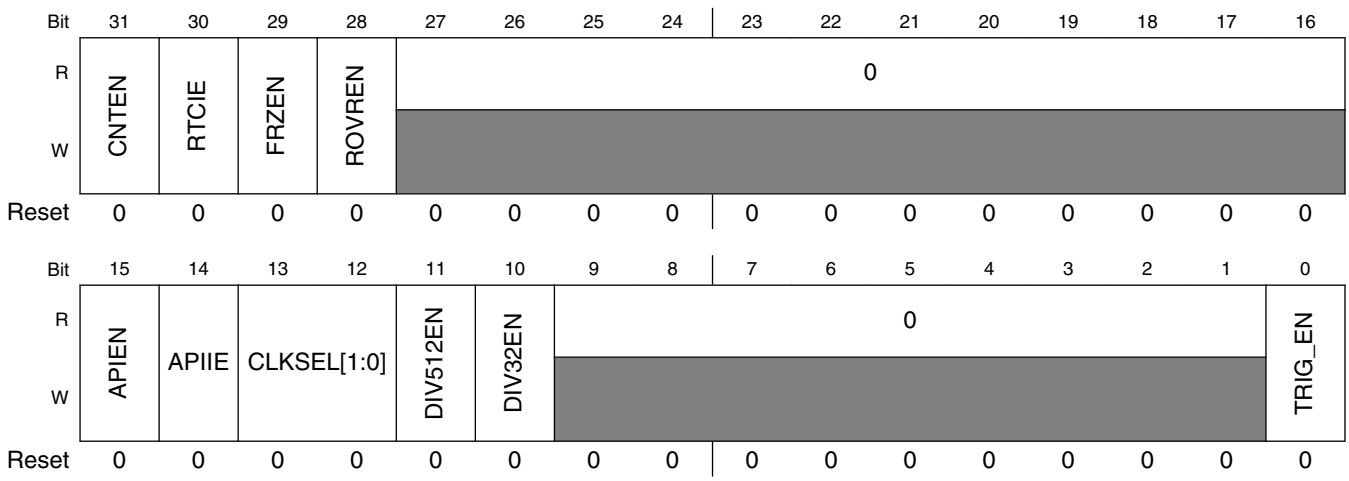
Table continues on the next page...

RTC_RTCSUPV field descriptions (continued)

Field	Description
	0 All registers are accessible in both user as well as supervisor mode 1 All other registers are accessible in supervisor mode only
Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

51.3.2 RTC Control register (RTC_RTCC)

Address: 0h base + 4h offset = 4h



RTC_RTCC field descriptions

Field	Description
31 CNTEN	Counter Enable. The CNTEN bit enable the RTC counter. Making CNTEN bit 1'b0 has the effect of asynchronously resetting (synchronous reset negation) all the RTC and API logic. This allows for the RTC configuration and clock source selection to be updated without causing synchronization issues. 0 Counter disabled. 1 Counter enabled.
30 RTCIE	RTC Interrupt Enable. The RTCIE bit enables interrupts requests to the system if RTCF is asserted. 0 RTC interrupts disabled. 1 RTC interrupts enabled.
29 FRZEN	Freeze Enable Bit. The counter freezes on entering the debug mode on the last valid count value if the FRZEN bit is set. After coming of the debug mode counter starts from the frozen value.

Table continues on the next page...

RTC_RTCC field descriptions (continued)

Field	Description
	0 Counter does not freeze in debug mode. 1 Counter freezes in debug mode.
28 ROVREN	Counter Roll Over Wakeup/Interrupt Enable. The ROVREN bit enables wakeup and interrupt requests when the RTC has rolled over from 0xffff_ffff to 0x0000_0000. The RTCIE bit must also be set in order to generate an interrupt from a counter rollover. 0 RTC rollover wakeup/interrupt disabled 1 RTC rollover wakeup/interrupt enabled.
27–16 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
15 APIEN	Autonomous Periodic Interrupt Enable. The APIEN bit enables the autonomous periodic interrupt function. 0 API disabled. 1 API enabled.
14 APIIE	API Interrupt Enable. The APIIE bit enables interrupts requests to the system if APIF is asserted. 0 API interrupts disabled. 1 API interrupts enabled.
13–12 CLKSEL[1:0]	Clock Select. The CLKSEL[1:0] bits select the clock source for the RTC. CLKSEL may only be updated when CNTEN is 0. The user should ensure that oscillator is enabled before selecting it as a clock source for RTC. 00 32 KHz (Slow)XOSC 01 128KHz (Slow)IRC 10 16MHz (Fast)IRC 11 8-40MHz (Fast)OSC
11 DIV512EN	Divide by 512 enable. The DIV512En bit enables the 512 clock divider. DIV512EN may only be updated when CNTEN is 0. 0 Divide by 512 is disabled. 1 Divide by 512 is enabled
10 DIV32EN	Divide by 32 enable. The DIV32En bit enables the 32 clock divider.DIV32EN may only be updated when CNTEN is 0. 0 Divide by 32 is disabled. 1 Divide by 32 is enabled.
9–1 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

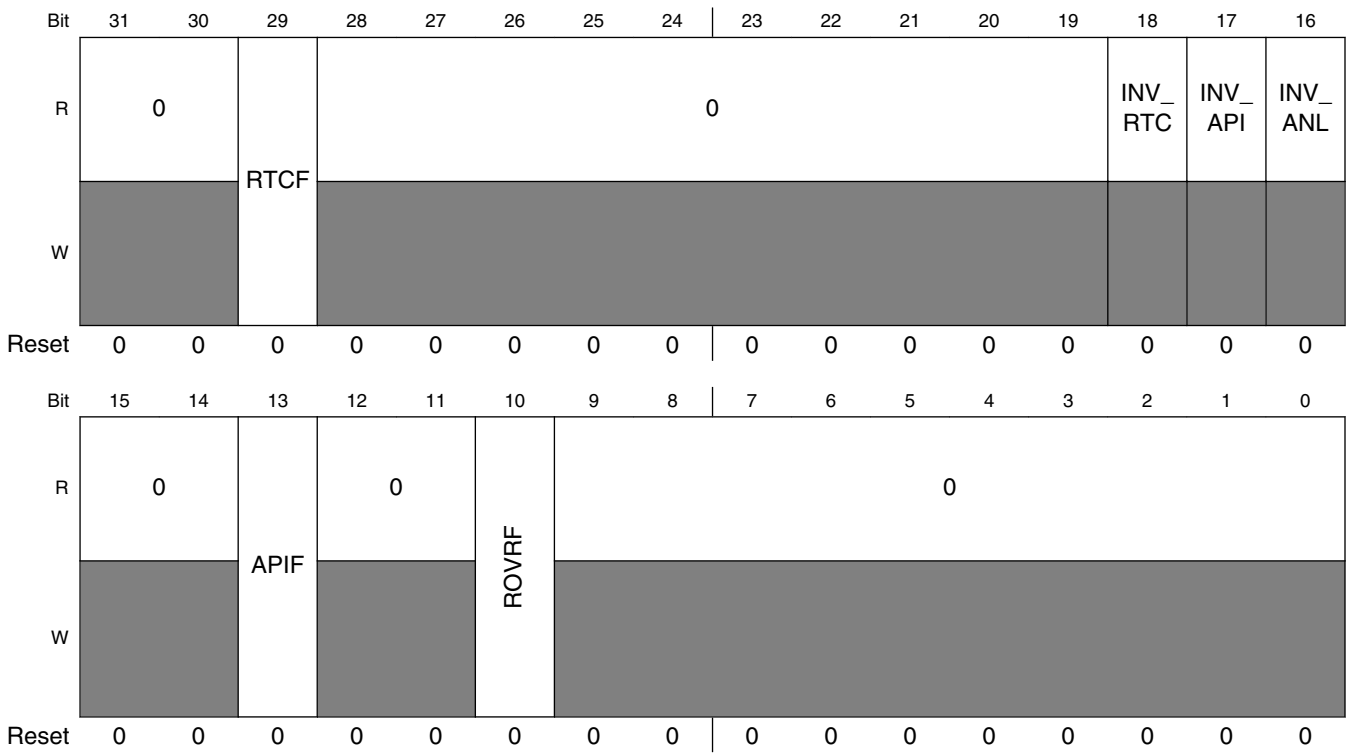
Table continues on the next page...

RTC_RTCC field descriptions (continued)

Field	Description
0 TRIG_EN	Trigger enable for Analog Comparator This bit when set will enable the trigger signal from RTC to Analog Comparator Block 0 Trigger enable is disabled 1 Trigger enable is enabled

51.3.3 RTC Status register (RTC_RTCS)

Address: 0h base + 8h offset = 8h



RTC_RTCS field descriptions

Field	Description
31–30 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
29 RTCF	RTC Interrupt Flag. The RTCF bit indicates that the RTC counter has reached the counter value matching RTCVAL. RTCF is cleared by writing a 1 to RTCF. Writing a 0 to RTCF has no effect. 0 RTC counter is not equal to RTCVAL. 1 RTC counter matches RTCVAL.

Table continues on the next page...

RTC_RTCS field descriptions (continued)

Field	Description
28–19 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
18 INV_RTC	Invalid rtc write This bit becomes value 1 after a value has been written to the RTCVAL register and the synchronization process is continuing. During this synchronization period, any attempt to write to the RTCVAL register again will be ignored.
17 INV_API	Invalid APIVAL write This bit becomes value 1 after a value has been written to the APIVAL register and the synchronization process is continuing. During this synchronization period, any attempt to write to the APIVAL register again will be ignored.
16 INV_ANL	Invalid ANLCMP write This bit becomes value 1 after a value has been written to the ANLCMP_CNT register and the synchronization process is continuing. During this synchronization period, any attempt to write to the ANLCMP_CNT register again will be ignored.
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 APIF	API Interrupt Flag. The APIF bit indicates that the RTC counter has reached the counter value matching API offset value. APIF is cleared by writing a 1 to APIF. Writing a 0 to APIF has no effect. 0 Counter is not equal to API offset value. 1 Counter matches the API offset value.
12–11 Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.
10 ROVRF	Counter Roll Over Interrupt Flag. The ROVRF bit indicates that the RTC has rolled over from 0xffff_ffff to 0x0000_0000. ROVRF is cleared by writing a 1 to ROVRF. 0 RTC has not rolled over 1 RTC has rolled over
Reserved	Reserved. This field is reserved. This read-only field is reserved and always has the value 0.

51.3.4 RTC Counter register (RTC_RTCCNT)

Address: 0h base + Ch offset = Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RTCCNT[31:0]																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_RTCCNT field descriptions

Field	Description
RTCCNT[31:0]	RTC Counter Value. Due to the clock synchronization, the RTCCNT value may actually represent a previous counter value.

51.3.5 API Compare Value register (RTC_APIVAL)

The APIVAL bits are compared to the RTC counter bits and if a match occurrs, an interrupt/wakeup request is asserted.

Address: 0h base + 10h offset = 10h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	APIVAL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_APIVAL field descriptions

Field	Description
APIVAL	API Compare Value. The APIVAL bits are compared to the RTC counter bits and if a match occurrs, an interrupt/wakeup request is asserted. **NOTE:** API functionality is active only when APIVAL is non zero. The first API interrupt takes two more cycles because of synchronization of APIVAL to rtc clock. After that, interrupts are periodic in nature and it takes APIVAL+1 cycles. Minimum supported value of APIVAL is 4. This is due to synchronization issues.

51.3.6 RTC Compare Value register (RTC_RTCVAL)

The RTCVAL bits are compared to the RTC counter bits and if a match occurs, RTCF is set. The minimum value of RTCVAL should be 4.

Address: 0h base + 14h offset = 14h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RTCVAL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_RTCVAL field descriptions

Field	Description
RTCVAL	RTC Compare Value. The RTCVAL bits are compared to the RTC counter bits and if a match occurs, RTCF is set.

51.3.7 RTC Analogue comparator count register (RTC_ANLCMP_CNT)

Address: 0h base + 18h offset = 18h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ANLCMP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RTC_ANLCMP_CNT field descriptions

Field	Description
ANLCMP	Analog Comparator Count Trigger from RTC to Analog Comparator block is generated as per the number of API clocks programmed into this register.

51.4 Functional Description

51.4.1 RTC

The RTC consists of a 32 bit free running counter enabled with the RTCC[CNTEN] bit (CNTEN when negated asynchronously resets the counter and synchronously enables the counter when enabled.) The value of the counter may be read via the RTCCNT register. Note that due to the clock synchronization, the RTCCNT value may actually represent a previous counter value. The difference between the counter and the read value depends on ratio of counter clock and Bus clock. Maximum possible difference between the two is 6 count values.

The clock source to the counter is selected with the RTCC[CLKSEL] field, which gives four options for clocking the RTC/API. The four clock sources are assumed to be one 16MHz source, one 32KHz source, one 128KHz source and one XOSC source. The output of the clock mux can be optionally divided by combination of 512 and 32 to give a 1ms RTC/API count period for different clock sources. Note that the RTCC[CNTEN] bit must be disabled when the RTC/API clock source is switched.

When the counter value for counter bits 31-0 match the 32-bit value in the RTCVAL[RTCVAL] field, then the RTCS[RTCF] interrupt flag bit is set (after proper clock synchronization). If the RTCC[RTCIE] interrupt enable bit is set, then the RTC interrupt request is generated. The RTC supports interrupt requests in the range of 1s to 4096s (> 1 hr.) with a 1s resolution. RTCVAL register can be written only when RTCS[INV_RTC] bit is clear. Initially RTCS[INV_RTC]=0, and hence RTCVAL can be written once and hence RTCS[INV_RTC] gets set. This bit can now be cleared only by enabling the RTC counter. Once the counter is enabled, RTCVAL can be written anytime now, untill RTC is disabled again. RTCVAL is first synchronized to rtc clock domain hence if RTCVAL is updated at the point where a counter match is due to happen in next two rtc clocks because of previous RTCVAL, RTIF flag will be set. However if the RTCVAL is updated at the point where no counter match is due as per the previous RTCVAL, the RTCF flag will be set when counter matches the new RTCVAL.

A rollover wakeup and/or interrupt can be generated when the RTC transitions from a count of 0xFFFF_FFFF to 0x0000_0000. The rollover flag is enabled by setting the RTCC[ROVREN] bit. An interrupt request is generated for an RTC counter rollover when both the RTCC[ROVREN] and RTCC[RTCIE] bits are set.

A trigger enable for the analog comparator block is also provided as an output from RTC API. A 32 bit counter starts incrementing if the RTCC[TRIG_EN] bit is set and the API asserts the wakeup signal. This counter will count up to the value programmed in the ANLCMP_CNT register and then will assert the trigger signal to Analog Comparator. This trigger is cleared on the next wakeup signal assertion.

All the flags and counter values are synchronized with Bus clock. It is assumed that the Bus clock frequency is always more than or equal to the RTC clock used to run the counter.

51.4.2 API functional description

Setting RTCC[APIEN] bit enables the autonomous interrupt function. The 32-bit RTCC[APIVAL] field selects the time interval for triggering an interrupt and/or wakeup event. Since the RTC is a free running counter, the APIVAL is added to the current count to calculate an offset. When the counter reaches the offset count, a interrupt and/or wakeup request is generated. Then the offset value is recalculated and again re-triggers a new request when the new value is reached. API function is enabled only when CNTEN and APIEN is asserted and APIVAL is non-zero. APIVAL can be updated anytime. After updation of APIVAL, the first API interrupt will be generated according the previous value. From the second interrupt onwards the API interrupt is generated with new APIVAL. When a compare is reached, the RTCS[APIF] interrupt flag bit is set (after proper clock synchronization). If the RTCC[APIIE] interrupt enable bit is set, then the API interrupt request is generated.

When the CNTEN is deasserted API function is reset.

Chapter 52
System Timer Module (STM)

52.1 Chip-specific STM information

52.1.1 STM configuration

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel.

The device includes two STMs, one for each core, with four 32-bit compare channels in each STM.

Table 52-1. System Timer Module (STM) instances

Instance	Description	Number of Channels	Number of Registers (32-bit)
STM_0	Intended for Z4 core in the high-speed computational clock domain.	4	14
STM_1	Intended for Z2 core in the peripheral clock domain.	4	14

52.1.1.1 STM Counter

In device STANDBY and LPU modes, only STM_0 is available.

NOTE

If any STM is enabled, then it should be re-initialized before entering into any low power mode (all STOP and STANDBY modes) so that the timer doesn't cause abort of the mode transitions.

52.1.2 STM register reset value

All register reset values for the STM registers that are device specific are shown below. All other registers reset values are shown in the STM Memory chapter

STM_0 Control register (SWT_0_TO) resets to 0x0000_ 0000h.

STM_1 Control register (SWT_1_TO) resets to 0x0000_ 0000h.

52.2 Introduction

This section provides an overview, list of features, and modes of operation for the STM.

52.2.1 Overview

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel.

52.2.2 Features

The STM has the following features:

- One 32-bit up counter with 8-bit prescaler
- Four 32-bit compare channels
- Independent interrupt source for each channel
- Counter can be stopped in debug mode
- Programmable selection of system clock or FXOSC clock for timer operation

52.2.3 Modes of operation

The STM supports two device modes of operation: normal and debug. When the STM is enabled in normal mode, its counter runs continuously. In debug mode, operation of the counter is controlled by the FRZ bit in the STM_CR register. If the FRZ bit is set, the counter is stopped in debug mode, otherwise it continues to run.

52.3 External signal description

The STM does not have any external interface signals.

52.4 Memory map and registers

The STM programming model includes a group of 32-bit registers—a module control register, a counter value register, and three registers for each channel.

The STM registers can only be accessed using 32-bit (word) accesses. Attempted references using a different size or to a reserved address generates a bus error termination.

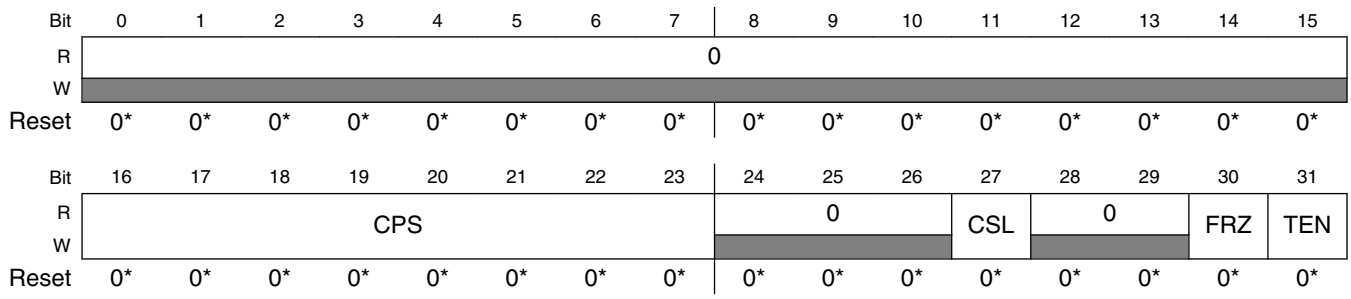
STM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	STM Control Register (STM_CR)	32	R/W	See section	52.4.1/2180
4	STM Count Register (STM_CNT)	32	R/W	0000_0000h	52.4.2/2181
10	STM Channel Control Register (STM_CCR0)	32	R/W	0000_0000h	52.4.3/2182
14	STM Channel Interrupt Register (STM_CIR0)	32	w1c	0000_0000h	52.4.4/2182
18	STM Channel Compare Register (STM_CMP0)	32	R/W	0000_0000h	52.4.5/2183
20	STM Channel Control Register (STM_CCR1)	32	R/W	0000_0000h	52.4.3/2182
24	STM Channel Interrupt Register (STM_CIR1)	32	w1c	0000_0000h	52.4.4/2182
28	STM Channel Compare Register (STM_CMP1)	32	R/W	0000_0000h	52.4.5/2183
30	STM Channel Control Register (STM_CCR2)	32	R/W	0000_0000h	52.4.3/2182
34	STM Channel Interrupt Register (STM_CIR2)	32	w1c	0000_0000h	52.4.4/2182
38	STM Channel Compare Register (STM_CMP2)	32	R/W	0000_0000h	52.4.5/2183
40	STM Channel Control Register (STM_CCR3)	32	R/W	0000_0000h	52.4.3/2182
44	STM Channel Interrupt Register (STM_CIR3)	32	w1c	0000_0000h	52.4.4/2182
48	STM Channel Compare Register (STM_CMP3)	32	R/W	0000_0000h	52.4.5/2183

52.4.1 STM Control Register (STM_CR)

The STM Control Register includes the prescale value, freeze control and timer enable bits.

Address: 0h base + 0h offset = 0h



* Notes:
- The reset value is implementation specific. See the Configuration information.

STM_CR field descriptions

Field	Description
0–15 Reserved	This read-only field is reserved and always has the value 0.
16–23 CPS	Counter Prescaler. Selects the clock divide value for the prescaler (1 - 256). 0x00 Divide system clock by 1 0x01 Divide system clock by 2 ---------- 0xFF Divide system clock by 256
24–26 Reserved	This read-only field is reserved and always has the value 0.
27 CSL	Clock Selection. Selects the clock that drives the internal timer. 0 System clock. 1 FXOSC clock.
28–29 Reserved	This read-only field is reserved and always has the value 0.
30 FRZ	Freeze. Allows the timer counter to be stopped when the device enters debug mode. **NOTE:** When the MCU enters debug mode, the STM is notified and uses the FRZ bit to determine counter mode.

Table continues on the next page...

STM_CR field descriptions (continued)

Field	Description
	0 STM counter continues to run in debug mode. 1 STM counter is stopped in debug mode.
31 TEN	Timer counter Enabled. 0 Counter is disabled. 1 Counter is enabled.

52.4.2 STM Count Register (STM_CNT)

The STM Count Register holds the timer count value.

NOTE

Due to two clock sources and different clock domains, Values in this field can lag behind the internal counter value for up to six system plus eight counter clock cycles. Therefore, the value read from this field immediately after may be lower than the actual value of the internal counter.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CNT field descriptions

Field	Description
0–31 CNT	Timer count value used as the time base for all channels. When enabled, the counter increments at the rate of the system clock divided by the prescale value.

52.4.3 STM Channel Control Register (STM_CCR*n*)

The STM Channel Control Register (STM_CCRn) has the enable bit for channel n of the timer.

Address: 0h base + 10h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															CEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CCR*n* field descriptions

Field	Description
0–30 Reserved	This read-only field is reserved and always has the value 0.
31 CEN	Channel Enable 0 The channel is disabled. 1 The channel is enabled.

52.4.4 STM Channel Interrupt Register (STM_CIR*n*)

The STM Channel Interrupt Register has the interrupt flag for channel n of the timer.

Address: 0h base + 14h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															CIF
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CIR*n* field descriptions

Field	Description
0–30 Reserved	This read-only field is reserved and always has the value 0.
31 CIF	Channel Interrupt Flag 0 No interrupt request. 1 Interrupt request due to a match on the channel.

52.4.5 STM Channel Compare Register (STM_CMP*n*)

The STM channel compare register (STM_CMPn) holds the compare value for channel n.

Address: 0h base + 18h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CMP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STM_CMP*n* field descriptions

Field	Description
0–31 CMP	Compare value for channel n. If the STM_CCRn[CEN] bit is set and the STM_CMPn register matches the STM_CNT register, a channel interrupt request is generated and the STM_CIRn[CIF] bit is set.

52.5 Functional description

The STM has one 32-bit up counter (STM_CNT) that is used as the time base for all channels. When enabled, the counter increments at the frequency of the clock selected by STM_CR[CSL] divided by a prescale value. The STM_CR[CPS] field sets the divider to any value in the range from 1 to 256. The counter is enabled with the STM_CR[TEN] bit. When enabled in normal mode the counter continuously increments. When enabled in debug mode the counter operation is controlled by the STM_CR[FRZ] bit. When the STM_CR[FRZ] bit is set, the counter is stopped in debug mode, otherwise it continues to run in debug mode. The counter rolls over at 0xFFFF_FFFF to 0x0000_0000 with no restrictions at this boundary.

The STM has four identical compare channels. Each channel includes a channel control register (STM_CCR*n*), a channel interrupt register (STM_CIR*n*), and a channel compare register (STM_CMP*n*). The channel is enabled by setting the STM_CCR*n*[CEN] bit. When enabled, the channel will set the STM_CIR*n*[CIF] bit and generate an interrupt request when the channel compare register matches the timer counter. The interrupt request is cleared by writing a 1 to the STM_CIR*n*[CIF] bit. A write of 0 to the STM_CIR*n*[CIF] bit has no effect.

Note

The STM counter does not advance when the system clock selected by STM_CR[CSL] is stopped.

Chapter 53
Reset Generation Module (MC_RGM)

53.1 Introduction

53.1.1 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the chip. It provides a register interface and the reset sequencer. Various registers are available to monitor and control the chip reset sequence. The reset sequencer is a state machine which controls the different phases (PHASE0, PHASE1, PHASE2, PHASE3, and IDLE) of the reset sequence and controls the reset signals generated in the system.

The following figure shows the MC_RGM block diagram.

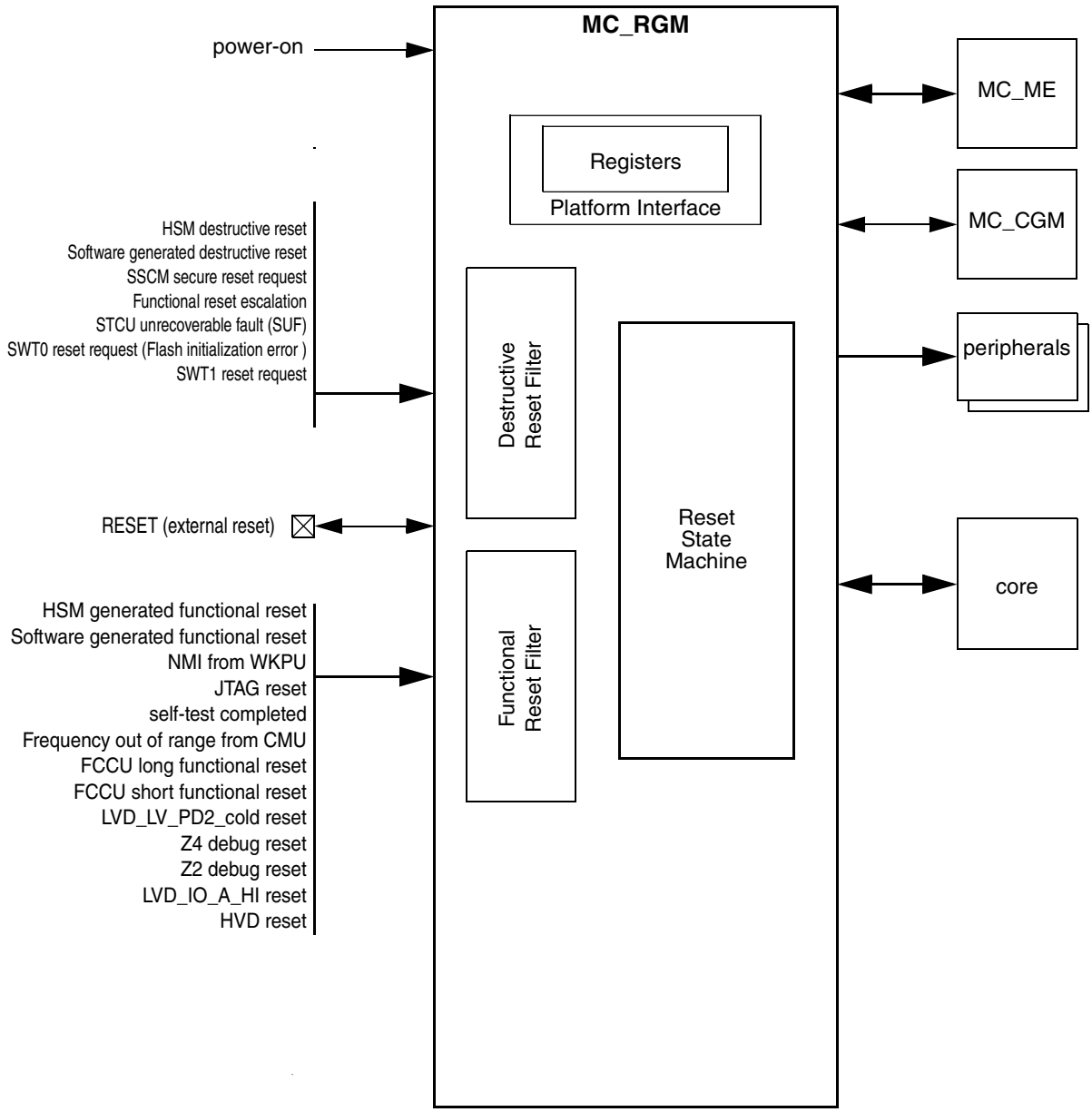


Figure 53-1. MC_RGM Block Diagram

53.1.2 Features

The MC_RGM contains the functionality for the following features:

- 'Destructive' resets management
- 'Functional' resets management
- Signaling of reset events after each reset sequence (reset status flags)
- Conversion of reset events to SAFE mode or interrupt request events

- short reset sequence configuration
- Bidirectional reset behavior configuration
- Boot mode capture on RESET deassertion
- Configurable escalation of recurring ‘functional’ resets to ‘destructive’ reset
- Configurable escalation of recurring ‘destructive’ resets to keep chip in reset state until next power-on reset

53.1.3 Reset Sources

The different reset sources are organized into two families: 'destructive' and 'functional'.

- A 'destructive' reset source is associated with an event related to a critical - usually hardware - error or dysfunction. When a 'destructive' reset event occurs, the full reset sequence is applied to the chip starting from PHASE0. This resets the full chip ensuring a safe start-up state for both digital and analog modules, and the memory content must be considered to be unknown. 'Destructive' resets are
  - power-on reset
  - HSM destructive reset
  - Software-generated destructive reset
  - SSCM secure reset request
  - 'functional' reset escalation
  - STCU unrecoverable fault (SUF)
  - SWT0 reset request (Flash initialization error)
  - SWT1 reset request
- A 'functional' reset source is associated with an event related to a less-critical - usually non-hardware - error or dysfunction. When a 'functional' reset event occurs, a partial reset sequence is applied to the chip starting from PHASE1. In this case, most digital modules are reset normally, while the state of analog modules or specific digital modules (e.g., debug modules, flash modules) as well as the system memory content is preserved. 'Functional' resets are
  - external reset
  - HSM-generated functional reset

- Software-generated functional reset
- NMI[3] from WKPU
- JTAG reset
- Self-test completed
- Frequency out of range from CMU
- FCCU long functional reset
- FCCU short functional reset
- LVD_LV_PD2_cold reset
- Z4A debug reset
- Z2 debug reset
- LVD_IO_A_HI reset
- HVD_LV_cold reset

When a reset is triggered, the MC_RGM state machine is activated and proceeds through the different phases (i.e., PHASEn states). Each phase is associated with a particular chip reset being provided to the system. A phase is completed when all corresponding phase completion gates from either the system or internal to the MC_RGM are acknowledged. The chip reset associated with the phase is then released, and the state machine proceeds to the next phase up to entering the IDLE phase. During this entire process, the MC_ME state machine is held in RESET mode. Only at the end of the reset sequence, when the IDLE phase is reached, does the MC_ME enter the DRUN mode.

Alternatively, it is possible for software to configure some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt issued to the core (see 'Functional' Event Reset Disable Register (MC_RGM_FERD) and 'Functional' Event Reset DisableAlternate Request Status Register (MC_RGM_FEAR)) for 'functional' resets.

53.2 External Signal Description

The MC_RGM interfaces to the bidirectional reset pin RESET.

53.3 Memory Map and Register Definition

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

Unless otherwise noted, all registers may be accessed as 32-bit words, 16-bit half-words, or 8-bit bytes. The bytes are ordered according to big endian. For example, the RGM_DES[8:15] register bits may be accessed as a word at address offset 0x000, as a half-word at address offset 0x002, or as a byte at address offset 0x003.

Some fields may be read-only, and their reset value of '1' or '0' and the corresponding behavior cannot be changed.

NOTE

The access defined for all the registers in the following table is for the User mode. For Supervisor and Test modes, please refer to the individual registers.

NOTE

Reserved fields should always be written with their reset values only.

NOTE

The following registers are reset to their default values during POR event only:

- RGM_DES
- RGM_FES
- RGM_DRET

MC_RGM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	'Destructive' Event Status Register (MC_RGM_DES)	32	R	0000_0001h	53.3.1/2190
30	'Destructive' Bidirectional Reset Enable Register (MC_RGM_DBRE)	32	R	0000_0000h	53.3.2/2192
300	'Functional' Event Status Register (MC_RGM_FES)	32	R	0000_0000h	53.3.3/2194
310	'Functional' Event Reset Disable Register (MC_RGM_FERD)	32	R	0000_8002h	53.3.4/2197

Table continues on the next page...

MC_RGM memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
320	'Functional' Event Reset DisableAlternate Request Status Register (MC_RGM_FEAR)	32	R	0000_0000h	53.3.5/2200
330	'Functional' Bidirectional Reset Enable Register (MC_RGM_FBRE)	32	R	0003_7FBCh	53.3.6/2202
340	'Functional' Event Short Sequence Register (MC_RGM_FESS)	32	R	0000_0204h	53.3.7/2205
604	'Functional' Reset Escalation Threshold Register (MC_RGM_FRET)	8	R	0Fh	53.3.8/2208
608	'Destructive' Reset Escalation Threshold Register (MC_RGM_DRET)	8	R	0Fh	53.3.9/2208

53.3.1 'Destructive' Event Status Register (MC_RGM_DES)

This register contains the status of the 'destructive' reset sources, including those configured to not generate a reset sequence. It can be accessed in read/write on either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on write '1'.

This register is reset only on power-on.

Access:

User read, Supervior read/write, Test read/write

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						0	F_SWT1_RES	F_SWT0_RES	F_SUF	F_FUNC_ESC	F_SSCM_SEC	F_SOFT_DEST	F_HSM_DEST	0	F_POR
W								w1c	w1c	w1c	w1c	w1c	w1c	w1c		w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MC_RGM_DES field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 F_SWT1_RES	Flag for SWT1 Reset 0 No SWT1 reset event has occurred since either the last clear or the last power-on reset assertion. 1 SWT1 reset event has occured.
24 F_SWT0_RES	Flag for SWT0 (Flash Initialization Failure) Reset 0 No SWT0 (Flash Initialization Failure) event has occurred since either the last clear or the last power-on reset assertion. 1 A SWT0 (Flash Initialization Failure) event has occurred.
25 F_SUF	Flag for SUF (STCU unrecoverable fault) 0 No SUF (STCU unrecoverable fault) event has occurred since either the last clear or the last power-on reset assertion. 1 A SUF (STCU unrecoverable fault) event has occurred.
26 F_FUNC_ESC	Flag for Functional Reset Escalation 0 No functional reset escalation event has occurred since either the last clear or the last power-on reset assertion. 1 A functional reset escalation event has occurred.
27 F_SSCM_SEC	Flag for SSCM Secure Reset Request 0 No SSCM Secure Reset Request event has occurred since either the last clear or the last power-on reset assertion. 1 An SSCM Secure Reset Request event has occurred.
28 F_SOFT_DEST	Flag for Software Generated Destructive Reset 0 No Software generated destructive reset event has occurred since either the last clear or the last power-on reset assertion. 1 A Software generated destructive reset event has occurred.

Table continues on the next page...

MC_RGM_DES field descriptions (continued)

Field	Description
29 F_HSM_DEST	Flag for HSM Destructive Reset 0 No HSM destructive reset event has occurred since either the last clear or the last power-on reset assertion. 1 A HSM destructive reset event has occurred.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 F_POR	Flag for Power-On Reset 0 No power-on event has occurred since the last clear. 1 A power-on event has occurred.

53.3.2 'Destructive' Bidirectional Reset Enable Register (MC_RGM_DBRE)

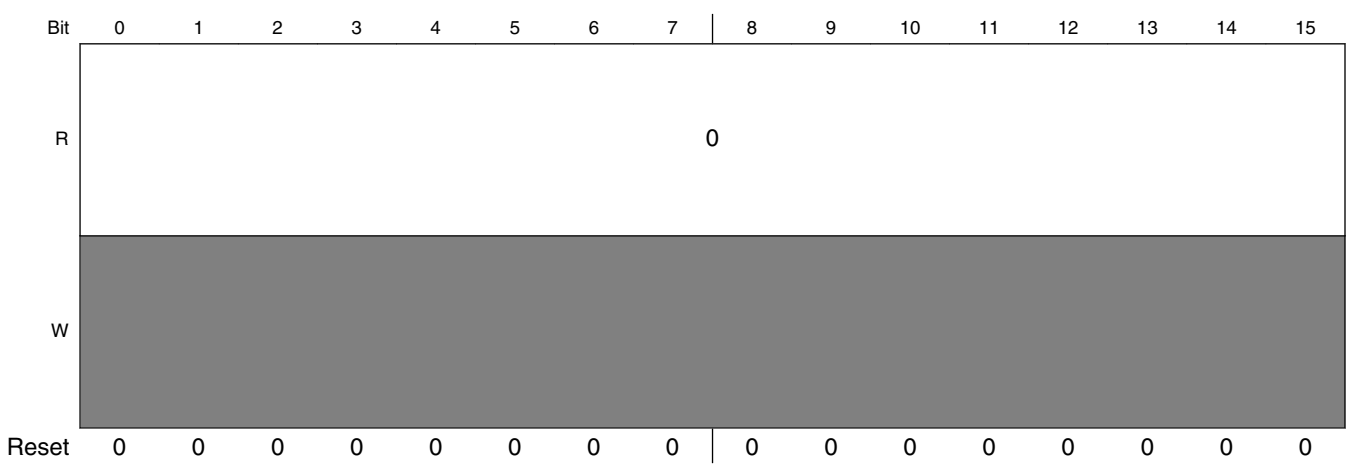
This register enables the generation of an external reset on ‘destructive’ reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

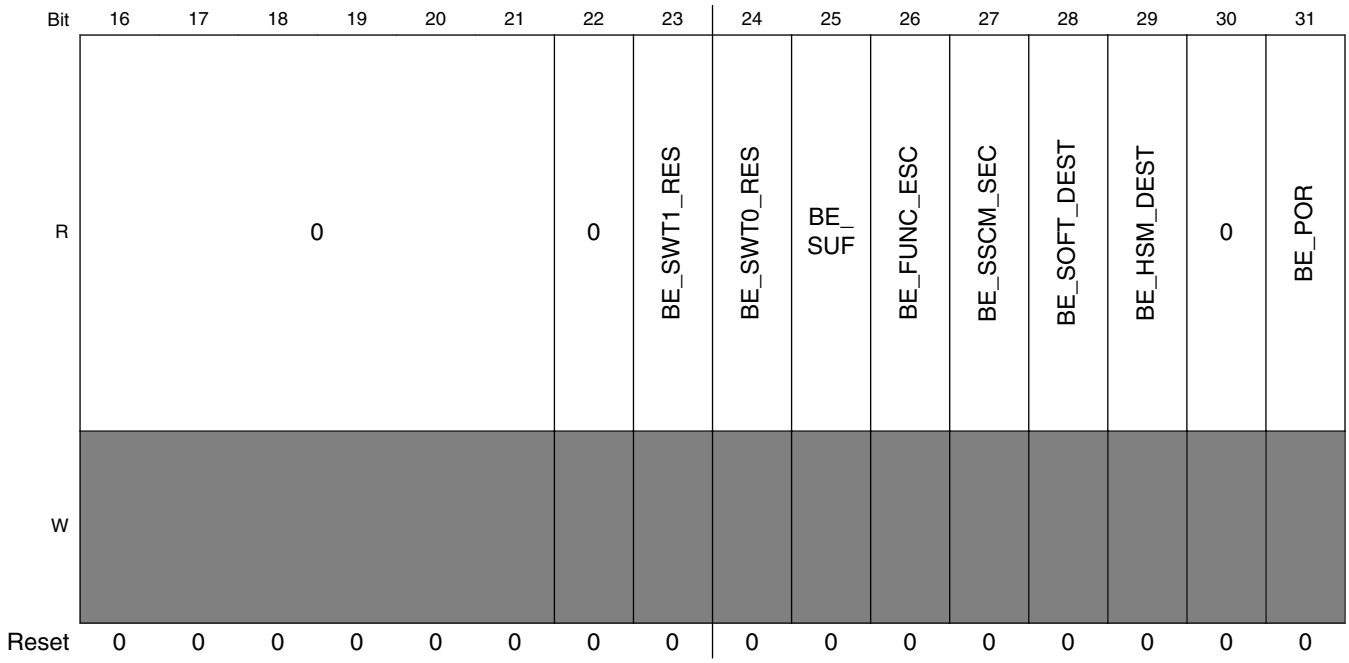
This register is reset only on power-on.

Access:

User read, Supervior read/write, Test read/write

Address: 0h base + 30h offset = 30h





MC_RGM_DBRE field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 BE_SWT1_RES	Bidirectional Reset Enable for SWT1 Reset 0 RESET is asserted on SWT1 Reset event if the reset is enabled.
24 BE_SWT0_RES	Bidirectional Reset Enable for SWT0 (Flash Initialization Failure) Reset 0 RESET is asserted on SWT0 (Flash Initialization Failure) Reset event if the reset is enabled.
25 BE_SUF	Bidirectional Reset Enable for SUF (STCU unrecoverable fault) 0 RESET is asserted on SUF (STCU unrecoverable fault) event if the reset is enabled.
26 BE_FUNC_ESC	Bidirectional Reset Enable for Functional Reset Escalation 0 RESET is asserted on a functional reset escalation event if the reset is enabled.
27 BE_SSCM_SEC	Bidirectional Reset Enable for SSCM Secure Reset Request 0 RESET is asserted on SSCM Secure Reset Request event if the reset is enabled.
28 BE_SOFT_DEST	Bidirectional Reset Enable for Software Generated Destructive Reset 0 RESET is asserted on a Software generated destructive reset event if the reset is enabled.
29 BE_HSM_DEST	Bidirectional Reset Enable for HSM Generated Functional Reset 0 RESET is asserted on a HSM genrated functional reset event if the reset is enabled.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_RGM_DBRE field descriptions (continued)

Field	Description
31 BE_POR	Bidirectional Power-On Reset 0 No power-on event has occurred since the last clear.

53.3.3 'Functional' Event Status Register (MC_RGM_FES)

This register contains the status of the ‘functional’ reset sources, including those configured to not generate a reset sequence. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Register bits are cleared on write ‘1’ if the triggering event has already been cleared at the source.

This register is reset only on power-on.

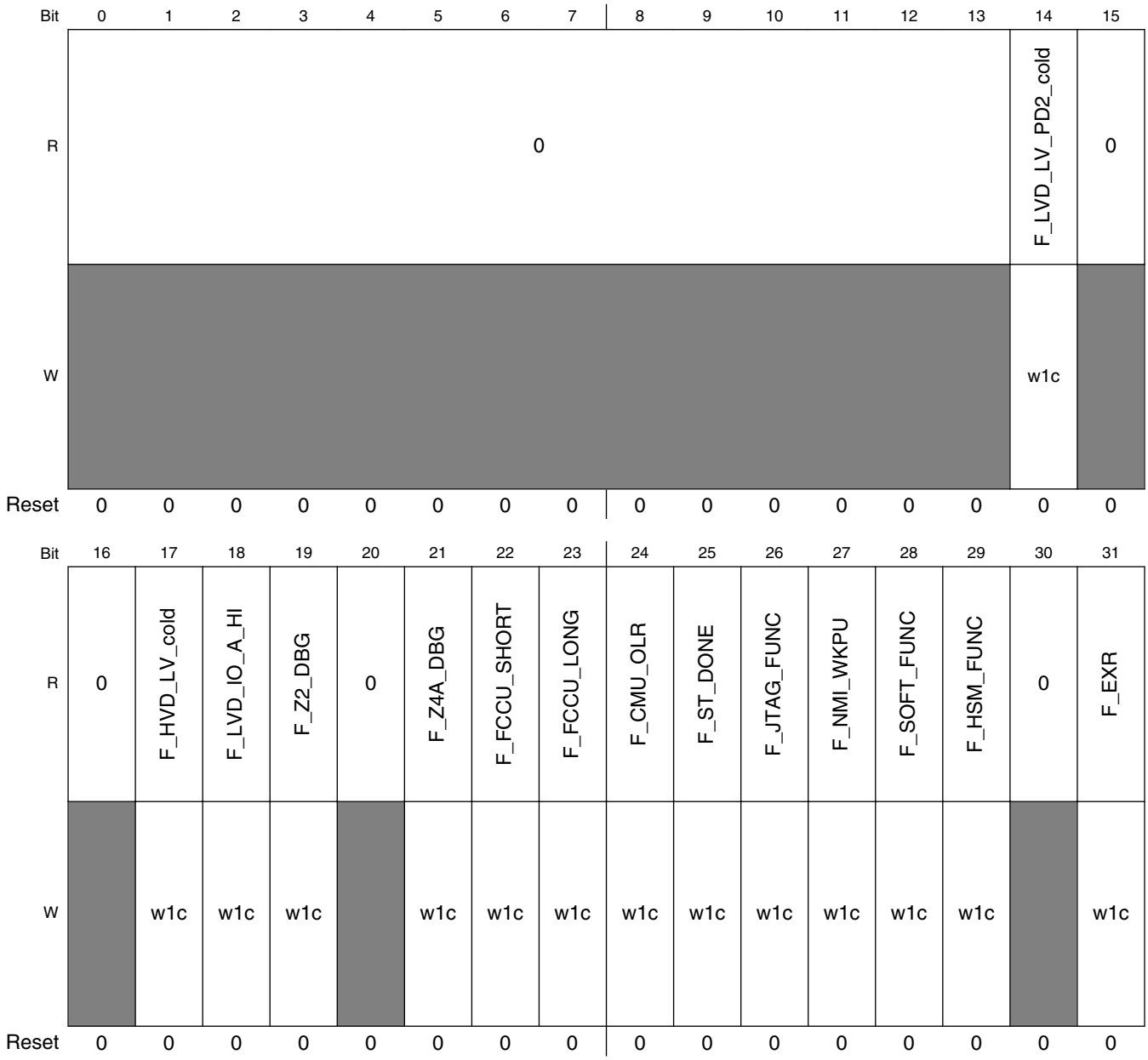
NOTE

If a ‘functional’ reset source is configured to generate a SAFE mode request or an interrupt request, software needs to clear the event in the source module at least three system clock cycles before it clears the associated RGM_FES status bit in order to avoid multiple SAFE mode requests or interrupts for the same event. In order to avoid having to count cycles, it is good practice for software to check whether the RGM_FES has been properly cleared, and if not, clear it again.

Access:

User read, Supervior read/write, Test read/write

Address: 0h base + 300h offset = 300h



MC_RGM_FES field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 F_LVD_LV_ PD2_cold	Flag for PD2 LVD at cold point 0 No PD2 LVD at cold point event has occurred since either the last clear or the last ‘destructive’ reset assertion. 1 A PD2 LVD at cold point event has occurred
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_RGM_FES field descriptions (continued)

Field	Description
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 F_HVD_LV_cold	Flag for High Voltage Detect 0 No High Voltage Detect event has occurred since either the last clear or the last ‘destructive’ reset assertion. 1 A High Voltage Detect event has occurred.
18 F_LVD_IO_A_HI	Flag for Low-Voltage Detect IOs 0 No Low-Voltage Detect IOs event has occurred since either the last clear or the last ‘destructive’ reset assertion. 1 Low-Voltage Detect IOs event has occurred.
19 F_Z2_DBG	Flag for Z2 Debug Reset 0 No Z2 Debug Reset event has occurred since either the last clear or the last ‘destructive’ reset assertion. 1 A Z2 Debug Reset event has occurred.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 F_Z4A_DBG	Flag for Z4A Debug Reset 0 No Z4A Debug Reset event has occurred since either the last clear or the last ‘destructive’ reset assertion. 1 A Z4A Debug Reset event has occurred.
22 F_FCCU_ SHORT	Flag for FCCU Short Functional Reset 0 No FCCU short functional reset event has occurred since either the last clear or the last ‘destructive’ reset assertion. 1 A FCCU short functional reset event has occurred.
23 F_FCCU_LONG	Flag for FCCU Long Functional Reset 0 No FCCU long functional reset event has occurred since either the last clear or the last ‘destructive’ reset assertion. 1 A FCCU long functional reset event has occurred.
24 F_CMU_OLR	Flag for OSC Frequency less than RC 0 No OSC Frequency less than RC event has occurred since either the last clear or the last ‘destructive’ reset assertion. 1 A OSC Frequency less than RC event has occurred.
25 F_ST_DONE	Flag for Selt Test Completed 0 No Selt test completed event has occurred since either the last clear or the last ‘destructive’ reset assertion. 1 A Selt test completed event has occurred.
26 F_JTAG_FUNC	Flag for JTAG Functional Reset 0 No JTAG Functional Reset event has occurred since either the last clear or the last ‘destructive’ reset assertion. 1 A JTAG Functional Reset event has occurred. For example, the EXTEST, HIGHZ, and CLAMP instructions cause a JTAG 'functional' reset event to occur, which sets this field to 1.

Table continues on the next page...

MC_RGM_FES field descriptions (continued)

Field	Description
27 F_NMI_WKPU	Flag for Non Maskable Interrupt from Wakeup Unit 0 No Non Maskable Interrupt from Wakeup Unit event has occurred since either the last clear or the last 'destructive' reset assertion. 1 A Non Maskable Interrupt from Wakeup Unit event has occurred.
28 F_SOFT_FUNC	Flag for Software Generated Functional Reset 0 No Software generated functional reset event has occurred since either the last clear or the last destructive' reset assertion. 1 A Software generated functional reset event has occurred.
29 F_HSM_FUNC	Flag for HSM Generated Functional Reset 0 No HSM genrated functional reset event has occurred since either the last clear or the last 'destructive' reset assertion. 1 A HSM genrated functional reset event has occurred.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 F_EXR	Flag for External Reset 0 No External reset event has occurred since either the last clear or the last 'destructive' reset assertion. 1 An external reset event has occurred.

53.3.4 'Functional' Event Reset Disable Register (MC_RGM_FERD)

This register provides dedicated bits to disable 'functional' reset sources.When a 'functional' reset source is disabled, the associated 'functional' event will trigger either a SAFE mode request or an interrupt request (see 'Functional' Event Reset DisableAlternate Request Status Register (MC_RGM_FEAR)). It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode. Each byte can be written only once after a 'destructive' or power-on reset.

This register is reset only on power-on and any 'destructive' reset.

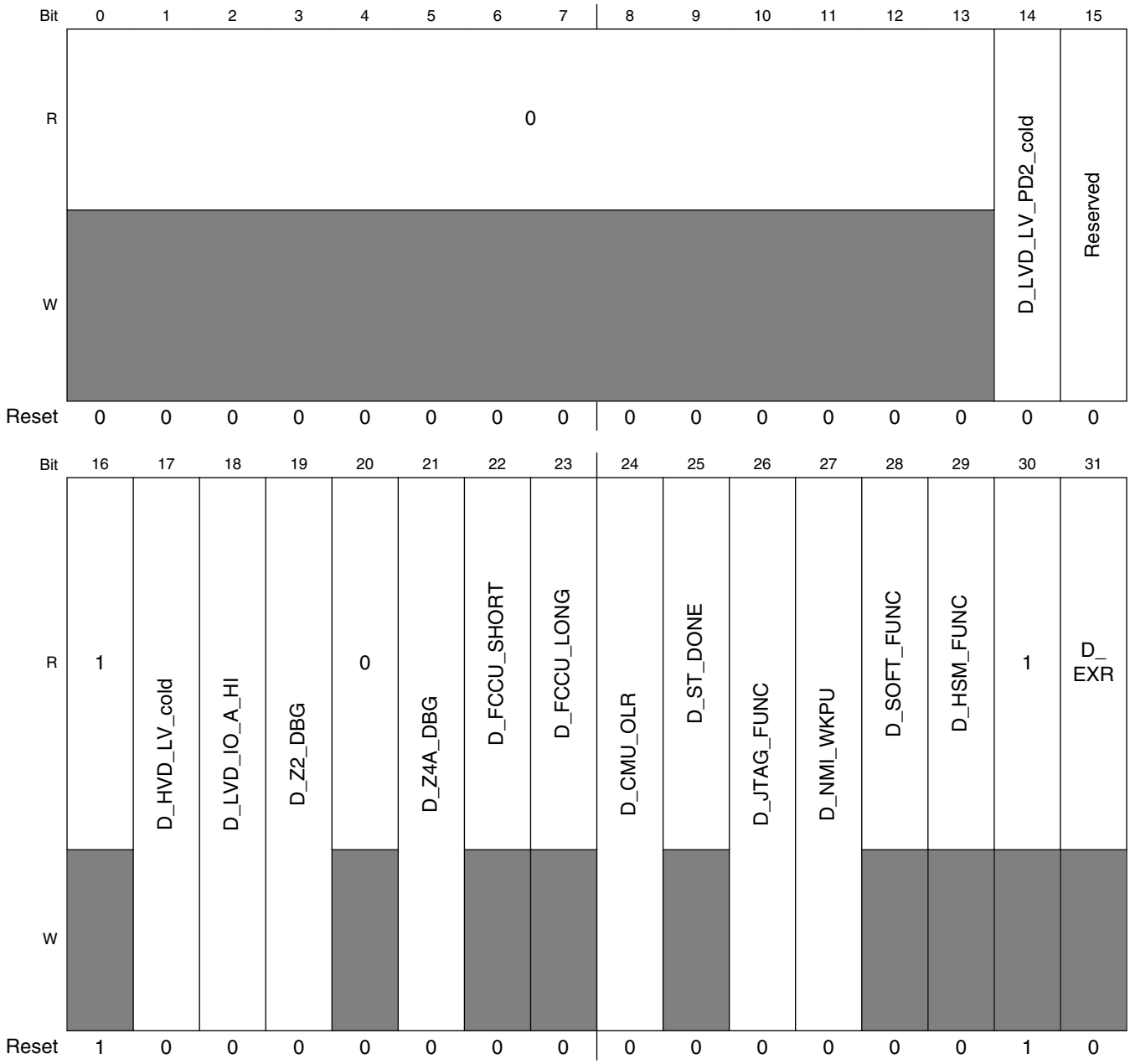
NOTE

It is important to clear the RGM_FES register before setting any of the bits in the RGM_FERD register to '1'. Otherwise a redundant SAFE mode request or interrupt request may occur.

Access:

User read, Supervior read/write, Test read/write

Address: 0h base + 310h offset = 310h



MC_RGM_FERD field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 D_LVD_LV_ PD2_cold	Disable PD2 LVD at cold point 0 A PD2 LVD at cold point event triggers a reset sequence. 1 A PD2 LVD at cold point event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_Z4A_DBG.
15 Reserved	This field is reserved.

Table continues on the next page...

MC_RGM_FERD field descriptions (continued)

Field	Description
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
17 D_HVD_LV_cold	Disable High Voltage Detect 0 A High Voltage Detect event triggers a reset sequence. 1 A High Voltage Detect event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_HVD_LV_cold.
18 D_LVD_IO_A_HI	Disable Low Voltage Detect IOs 0 A Low Voltage Detect IOs event triggers a reset sequence. 1 A Low Voltage Detect IOs event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_LVD_IO.
19 D_Z2_DBG	Disable Z2 Debug Rreset 0 A Z2 Debug Reset event triggers a reset sequence. 1 A Z2 Debug Reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_Z2_DBG.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 D_Z4A_DBG	Disable Z4A Debug Reset 0 A Z4A Debug Reset event triggers a reset sequence. 1 A Z4A Debug Reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_Z4A_DBG.
22 D_FCCU_ SHORT	Disable FCCU Short Functional Reset 0 A FCCU short functional reset event triggers a reset sequence.
23 D_FCCU_LONG	Disable FCCU Long Functional Reset 0 A FCCU long functional reset event triggers a reset sequence.
24 D_CMU_OLR	Disable OSC Frequency less than RC **NOTE** In case OLR is configured as safe mode request, Exit from SAFE mode should be always done via any functional Reset to avoid any unpredictable behavior. 0 OSC Frequency less than RC event triggers a reset sequence. 1 OSC Frequency less than RC event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_CMU_OLR.
25 D_ST_DONE	Disable Selt Test Completed 0 A Selt test completed event triggers a reset sequence.
26 D_JTAG_FUNC	Disable JTAG Functional Reset 0 JTAG Functional Reset event triggers a reset sequence. 1 JTAG Functional Reset event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_JTAG_FUNC.
27 D_NMI_WKPU	Disable Non Maskable Interrupt from Wakeup Unit

Table continues on the next page...

MC_RGM_FERD field descriptions (continued)

Field	Description
	0 A Non Maskable Interrupt from Wakeup Unit event triggers a reset sequence. 1 A Non Maskable Interrupt from Wakeup Unit event generates either a SAFE mode or an interrupt request depending on the value of RGM_FEAR.AR_NMI_WKPU.
28 D_SOFT_FUNC	Disable Software Generated Functional Reset 0 A Software generated functional reset event triggers a reset sequence.
29 D_HSM_FUNC	Disable HSM Genrated Functional Reset 0 A HSM genrated functional reset event triggers a reset sequence
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
31 D_EXR	Disable External Reset 0 No External reset event has occurred since either the last clear or the last ‘destructive’ reset assertion.

53.3.5 'Functional' Event Reset DisableAlternate Request Status Register (MC_RGM_FEAR)

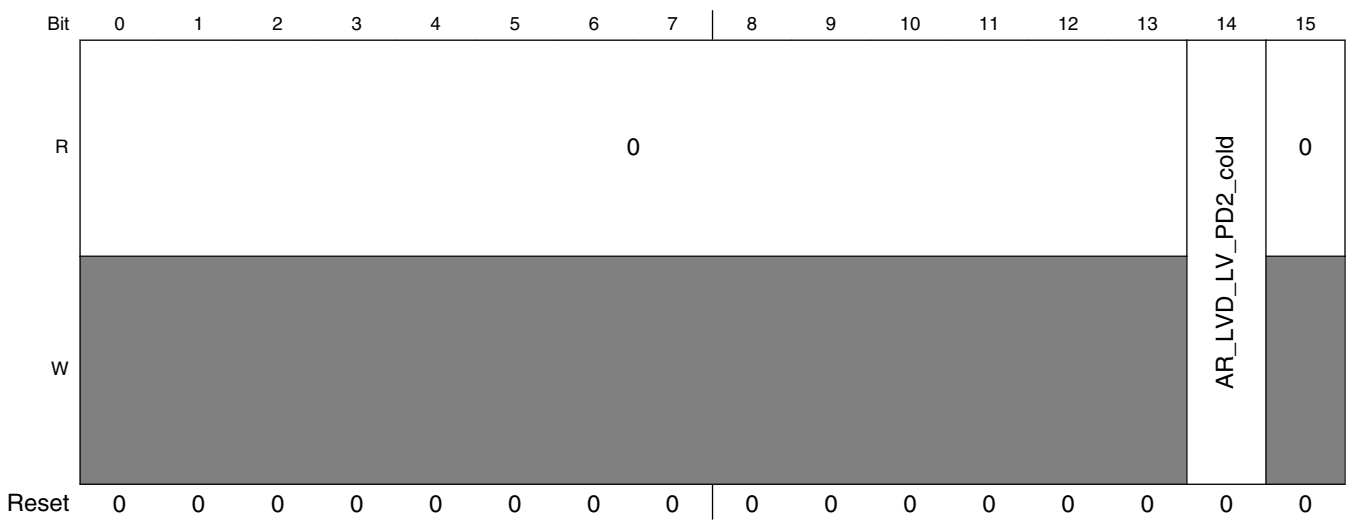
This register defines an alternate request to be generated when a reset on a ‘functional’ event has been disabled. The alternate request can be either a SAFE mode request to MC_ME or an interrupt request to the system. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

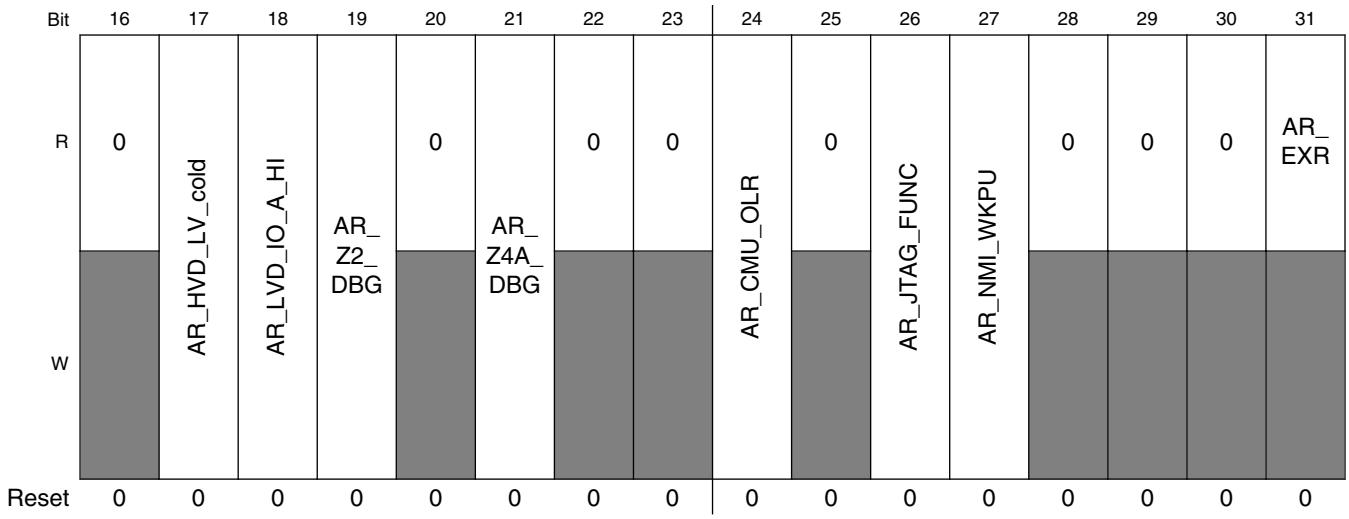
This register is reset only on power-on and any ‘destructive’ reset.

Access:

User read, Supervior read/write, Test read/write

Address: 0h base + 320h offset = 320h





MC_RGM_FEAR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 AR_LVD_LV_PD2_cold	Alternate Request for PD2 LVD at cold point 0 Generate a SAFE mode request on PD2 LVD at cold point event if the reset is disabled. 1 Generate an interrupt request on PD2 LVD at cold point event if the reset is disabled.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 AR_HVD_LV_cold	Alternate Request for High Voltage Detect 0 Generate a SAFE mode request on a High Voltage Detect event if the reset is disabled. 1 Generate an interrupt request on High Voltage Detect event if the reset is disabled.
18 AR_LVD_IO_A_HI	Alternate Request for Low Voltage Detect IOs 0 Generate a SAFE mode request on Low Voltage Detect IOs event if the reset is disabled. 1 Generate an interrupt request on Low Voltage Detect IOs event if the reset is disabled.
19 AR_Z2_DBG	Alternate Request for Z2 Debug Reset 0 Generate a SAFE mode request on Z2 Debug Reset event if the reset is disabled. 1 Generate an interrupt request on Z2 Debug Reset event if the reset is disabled.
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 AR_Z4A_DBG	Alternate Request for Z4A Debug Reset 0 Generate a SAFE mode request on Z4A Debug Reset event if the reset is disabled. 1 Generate an interrupt request on ZA Debug Reset event if the reset is disabled.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

MC_RGM_FEAR field descriptions (continued)

Field	Description
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 AR_CMU_OLR	Alternate Request for OSC Frequency less than RC **NOTE:** In case OLR is configured as safe mode request, Exit from SAFE mode should be always done via any functional Reset to avoid any unpredictable behavior. 0 Generate a SAFE mode request on a OSC Frequency less than RC event if the reset is disabled. 1 Generate an interrupt request on a OSC Frequency less than RC event if the reset is disabled.
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 AR_JTAG_FUNC	Alternate Request for JTAG functional Reset 0 Generate a SAFE mode request on a JTAG functional Reset event if the reset is disabled. 1 Generate an interrupt request on a JTAG functional Reset event if the reset is disabled.
27 AR_NMI_WKPU	Alternate Request for Non Maskable Interrupt from Wakeup Unit 0 Generate a SAFE mode request on a Non Maskable Interrupt from Wakeup Unit event if the reset is disabled.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0. .
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 AR_EXR	Flag for External reset 0 Generate a SAFE mode request on an External reset event if the reset is disabled.

53.3.6 'Functional' Bidirectional Reset Enable Register (MC_RGM_FBRE)

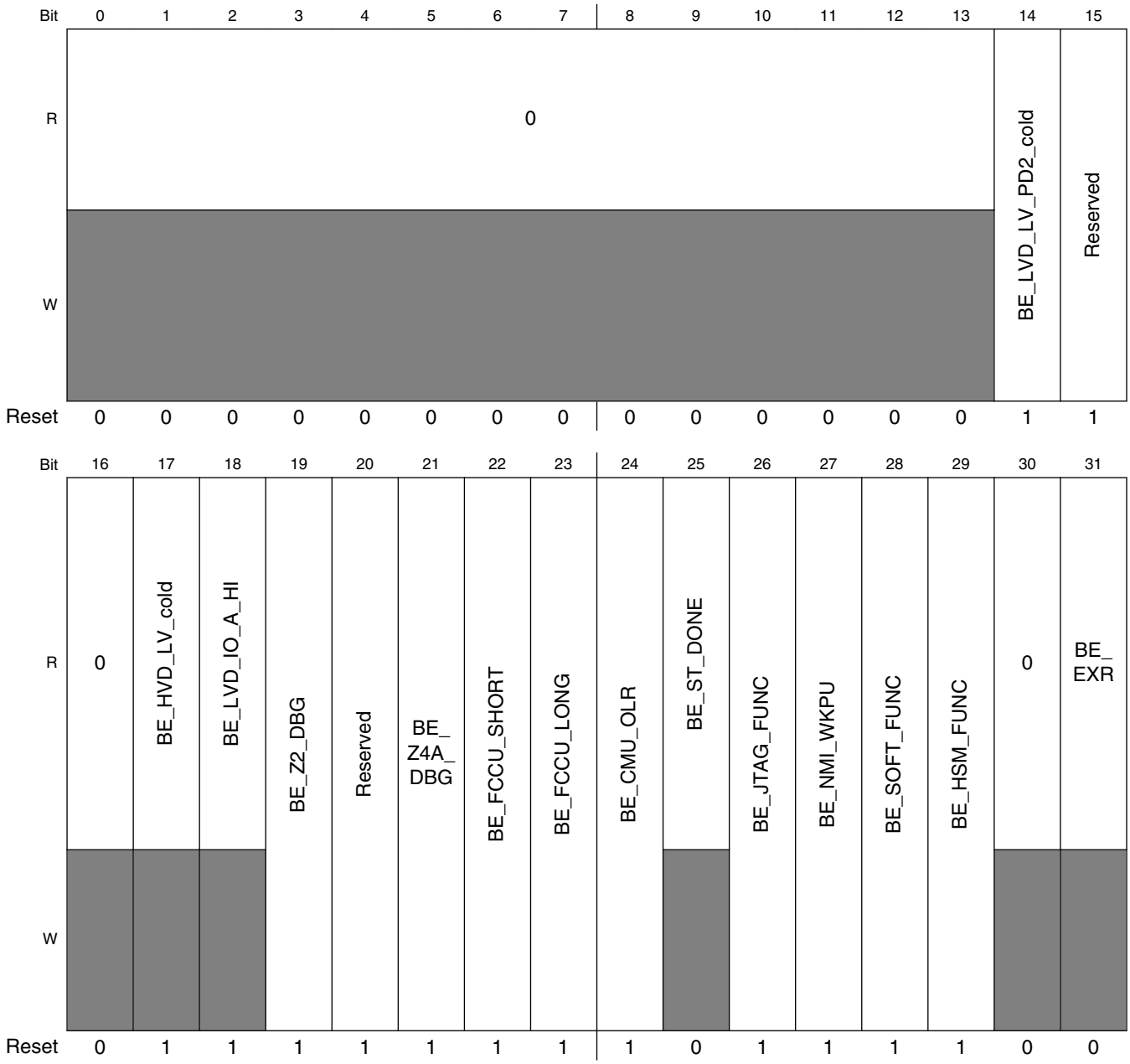
This register enables the generation of an external reset on ‘functional’ reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode..

This register is reset only on power-on and any ‘destructive’ reset.

Access:

User read, Supervior read/write, Test read/write

Address: 0h base + 330h offset = 330h



MC_RGM_FBRE field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 BE_LVD_LV_ PD2_cold	Bidirectional Reset Enable for PD2 LVD at Cold Point 0 RESET is asserted on a PD2 LVD at cold point event if the reset is enabled. 1 RESET is not asserted on a PD2 LVD at cold point event.
15 Reserved	This field is reserved.

Table continues on the next page...

MC_RGM_FBRE field descriptions (continued)

Field	Description
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 BE_HVD_LV_cold	Bidirectional Reset Enable for High Voltage Detect 0 RESET is asserted on a High Voltage Detect event if the reset is enabled. 1 RESET is not asserted on a High Voltage Detect event.
18 BE_LVD_IO_A_HI	Bidirectional Reset Enable for Low Voltage Detect IOs 0 RESET is asserted on a Low Voltage Detect IOs event if the reset is enabled. 1 RESET is not asserted on a Low Voltage Detect IOs event.
19 BE_Z2_DBG	Bidirectional Reset Enable for Z2 debug reset 0 RESET is asserted on a Z2 debug reset event if the reset is enabled. 1 RESET is not asserted on a Z2 debug reset event.
20 Reserved	This field is reserved.
21 BE_Z4A_DBG	Bidirectional Reset Enable for Z4A Debug Reset 0 RESET is asserted on a Z4A debug reset event if the reset is enabled. 1 RESET is not asserted on a Z4A debug reset event.
22 BE_FCCU_SHORT	Bidirectional Reset Enable for FCCU Short Functional Reset 0 RESET is asserted on a FCCU short functional reset event if the reset is enabled. 1 RESET is not asserted on a FCCU Short Functional Reset event.
23 BE_FCCU_LONG	Bidirectional Reset Enable for FCCU Long Functional Reset 0 RESET is asserted on a FCCU long functional reset event if the reset is enabled. 1 RESET is not asserted on a FCCU long functional reset event.
24 BE_CMU_OLR	Bidirectional Reset Enable for OSC Frequency less than RC 0 RESET is asserted on an OSC Frequency less than RC event if the reset is enabled. 1 RESET is not asserted on an OSC Frequency less than RC event.
25 BE_ST_DONE	Bidirectional Reset Enable for Selt Test Completed This read-only field always has value 0. 0 RESET is asserted on Selt Test Completed event if the reset is enabled.
26 BE_JTAG_FUNC	Bidirectional Reset Enable for JTAG Function 0 RESET is asserted on a JTAG Function event if the reset is enabled.. 1 RESET is not asserted on a JTAG Function event.
27 BE_NMI_WKPU	Bidirectional Reset Enable for Non Maskable Interrupt from Wakeup Unit 0 RESET is asserted on Non Maskable Interrupt from Wakeup Unit event if the reset is enabled. 1 RESET is not asserted on Non Maskable Interrupt from Wakeup Unit event.
28 BE_SOFT_FUNC	Bidirectional Reset Enable for Software Generated Functional Reset 0 RESET is asserted on a Software generated destructive reset event if the reset is enabled. 1 RESET is not asserted on a Software generated destructive reset event.

Table continues on the next page...

MC_RGM_FBRE field descriptions (continued)

Field	Description
29 BE_HSM_FUNC	Bidirectional Reset Enable for HSM Generated Functional Reset 0 RESET is asserted on a HSM genrated functional reset event if the reset is enabled. 1 RESET is not asserted on a HSM genrated functional reset event.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 BE_EXR	Flag for External Reset 0 RESET is asserted on External reset event if the reset is enabled.

53.3.7 'Functional' Event Short Sequence Register (MC_RGM_FESS)

This register defines which reset sequence will be done when a ‘functional’ reset sequence is triggered. The ‘functional’ reset sequence can either start from PHASE1 or from PHASE3, skipping PHASE1 and PHASE2.

NOTE

This could be useful for a fast reset sequence, for example to skip flash reset.

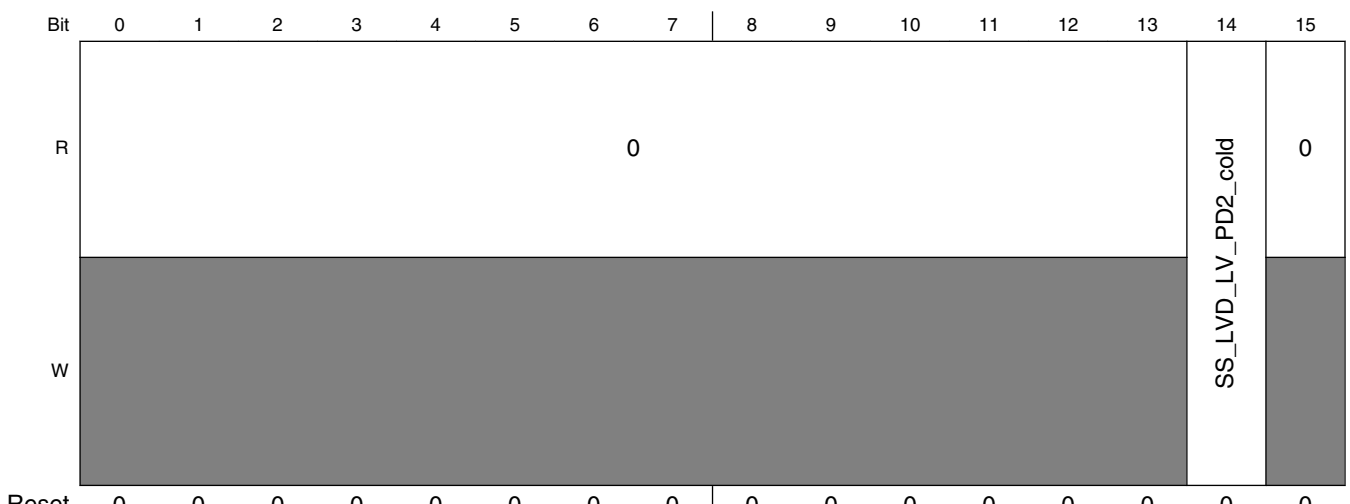
It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read in user mode.

This register is reset only on power-on and any ‘destructive’ reset.

Access:

User read, Supervior read/write, Test read/write

Address: 0h base + 340h offset = 340h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	SS_HVD_LV_cold	SS_LVD_IO_A_HI	SS_Z2_DBG	0	SS_Z4A_DBG	SS_FCCU_SHORT	SS_FCCU_LONG	SS_CMU_OLR	SS_ST_DONE	SS_FUNC_JTAG	SS_NMI_WKPU	SS_SOFT_FUNC	SS_HSM_FUNC	0	SS_EXR
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0

MC_RGM_FESS field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 SS_LVD_LV_PD2_cold	Short Sequence for PD2 LVD at Cold Point 0 The reset sequence triggered by a PD2 LVD at Cold Point event will start from PHASE1. 1 The reset sequence triggered by a PD2 LVD at Cold Point event will start from PHASE3, skipping PHASE1 and PHASE2.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 SS_HVD_LV_cold	Short Sequence for High Voltage Detect IOs 0 The reset sequence triggered by a high Voltage Detect IOs event will start from PHASE1. 1 The reset sequence triggered by a high Voltage Detect IOs event will start from PHASE3, skipping PHASE1 and PHASE2.
18 SS_LVD_IO_A_HI	Short Sequence for Low Voltage Detect IOs 0 The reset sequence triggered by a Low Voltage Detect IOs event will start from PHASE1. 1 The reset sequence triggered by a Low Voltage Detect IOs event will start from PHASE3, skipping PHASE1 and PHASE2.
19 SS_Z2_DBG	Short Sequence for Z2 Debug Reset 0 The reset sequence triggered by a Z2 Debug Reset event will start from PHASE1. 1 The reset sequence triggered by a Z2 Debug Reset event will start from PHASE3, skipping PHASE1 and PHASE2.

Table continues on the next page...

MC_RGM_FESS field descriptions (continued)

Field	Description
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 SS_Z4A_DBG	Short Sequence for Z4A Debug Reset 0 The reset sequence triggered by a Z4A Debug Reset event will start from PHASE1. 1 The reset sequence triggered by a Z4A Debug Reset event will start from PHASE3, skipping PHASE1 and PHASE2.
22 SS_FCCU_ SHORT	Short Sequence for FCCU Short Functional Reset 1 The reset sequence triggered by a FCCU short functional reset event will start from PHASE3, skipping PHASE1 and PHASE2.
23 SS_FCCU_ LONG	Short Sequence for FCCU Long Functional Reset 0 The reset sequence triggered by a FCCU long functional reset event will start from PHASE1.
24 SS_CMU_OLR	Short Sequence for OSC Frequency less than RC 0 The reset sequence triggered by a OSC Frequency less than RC event will start from PHASE1. 1 The reset sequence triggered by OSC Frequency less than RC event will start from PHASE3, skipping PHASE1 and PHASE2.
25 SS_ST_DONE	Short Sequence for Selt Test Completed 0 The reset sequence triggered by a Selt test completed event will start from PHASE1.
26 SS_FUNC_JTAG	Short Sequence for JTAG Functional Reset 0 The reset sequence triggered by a JTAG Functional Reset event will start from PHASE1. 1 The reset sequence triggered by a JTAG Functional Reset event will start from PHASE3, skipping PHASE1 and PHASE2.
27 SS_NMI_WKPU	Short Sequence for Non Maskable Interrupt 3 Wakeup Unit 0 The reset sequence triggered by a Non Maskable Interrupt Wakeup Unit event will start from PHASE1. 1 The reset sequence triggered by a Non Maskable Interrupt 3 Wakeup Unit event will start from PHASE3, skipping PHASE1 and PHASE2.
28 SS_SOFT_FUNC	Short Sequence for Software Generated Functional Reset 0 The reset sequence triggered by a Software generated functional reset event will start from PHASE1. 1 The reset sequence triggered by a Software generated functional reset event will start from PHASE3, skipping PHASE1 and PHASE2.
29 SS_HSM_FUNC	Short Sequence for HSM Genrated Functional Reset 1 The reset sequence triggered by a HSM genrated functional reset event will start from PHASE3, skipping PHASE1 and PHASE2.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 SS_EXR	Short Sequence for External Reset 0 The reset sequence triggered by an External reset reset event will start from PHASE1. 1 The reset sequence triggered by an External reset reset event will start from PHASE3, skipping PHASE1 and PHASE2.

53.3.8 'Functional' Reset Escalation Threshold Register (MC_RGM_FRET)

This register sets the threshold for ‘functional’ reset escalation to a ‘destructive’ reset. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

Writing a non-zero value to the FRET field will enable the ‘functional’ reset escalation function. Writing any value to this register will reset the ‘functional’ reset counter. See Functional’ Reset Escalation for details on the ‘functional’ reset escalation function.

This register is reset only on power-on and any ‘destructive’ reset.

Access:

User read, Supervior read/write, Test read/write

Address: 0h base + 604h offset = 604h

Bit	0	1	2	3	4	5	6	7
Read	0				FRET			
Write								
Reset	0	0	0	0	1	1	1	1

MC_RGM_FRET field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 FRET	‘Functional’ Reset Escalation Threshold — If the value of this field is 0, the ‘functional’ reset escalation function is disabled. Any other value is the number of ‘functional’ resets which will cause a ‘destructive’ reset if the RGM_FRET register isn’t written to beforehand.

53.3.9 'Destructive' Reset Escalation Threshold Register (MC_RGM_DRET)

This register sets the threshold for ‘destructive’ reset escalation to keeping the chip in the reset state until the next power-on reset triggers a new reset sequence. It can be accessed in read/write in either supervisor mode or test mode. It can be accessed in read only in user mode.

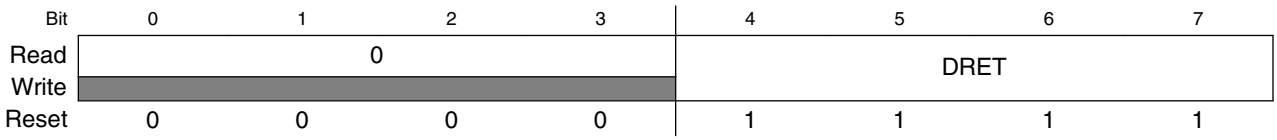
Writing a non-zero value to the DRET field will enable the ‘destructive’ reset escalation function. Writing any value to this register will reset the ‘destructive’ reset counter. See Destructive’ Reset Escalation” for details on the ‘destructive’ reset escalation function.

This register is reset only on power-on.

Access:

User read, Supervior read/write, Test read/write

Address: 0h base + 608h offset = 608h



MC_RGM_DRET field descriptions

Field	Description
0–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 DRET	'Destructive' Reset Escalation Threshold — If the value of this field is 0, the 'destructive' reset escalation function is disabled. Any other value is the number of 'destructive' resets which will keep the chip in the reset state until the next power-on reset triggers a new reset sequence if the RGM_DRET register isn't written to beforehand.

53.4 Functional Description

53.4.1 Reset State Machine

The main role of the MC_RGM is the generation of the reset sequence which ensures that the correct parts of the chip are reset based on the reset source event. This is summarized in the following table.

Table 53-1. MC_RGM Reset Implications

Source	What Gets Reset	External Reset Assertion[1]
power-on reset	all	yes
'destructive' resets	all except some clock/reset management	programmable[2]
external reset	all except some clock/reset management and debug logic	programmable
'functional' resets	all except some clock/reset management and debug logic	programmable[3]
shortened 'functional' resets[4]	flip-flops except some clock/reset management and flash	programmable[3]

1. 'External reset assertion' means that the RESET (external reset) pin is asserted by the MC_RGM from the start of the reset sequence until the end of reset PHASE3.
2. The assertion of the external reset is controlled via the RGM_DBRE register.
3. The assertion of the external reset is controlled via the RGM_FBRE register.
4. The short sequence is enabled via the RGM_FESS register.

Note

JTAG logic has its own independent reset control and is not controlled by the MC_RGM in any way.

The reset sequence is comprised of five phases managed by a state machine, which ensures that all phases are correctly processed through waiting for a minimum duration and until all processes that need to occur during that phase have been completed before proceeding to the next phase.

The state machine used to produce the reset sequence is shown in the following figure.

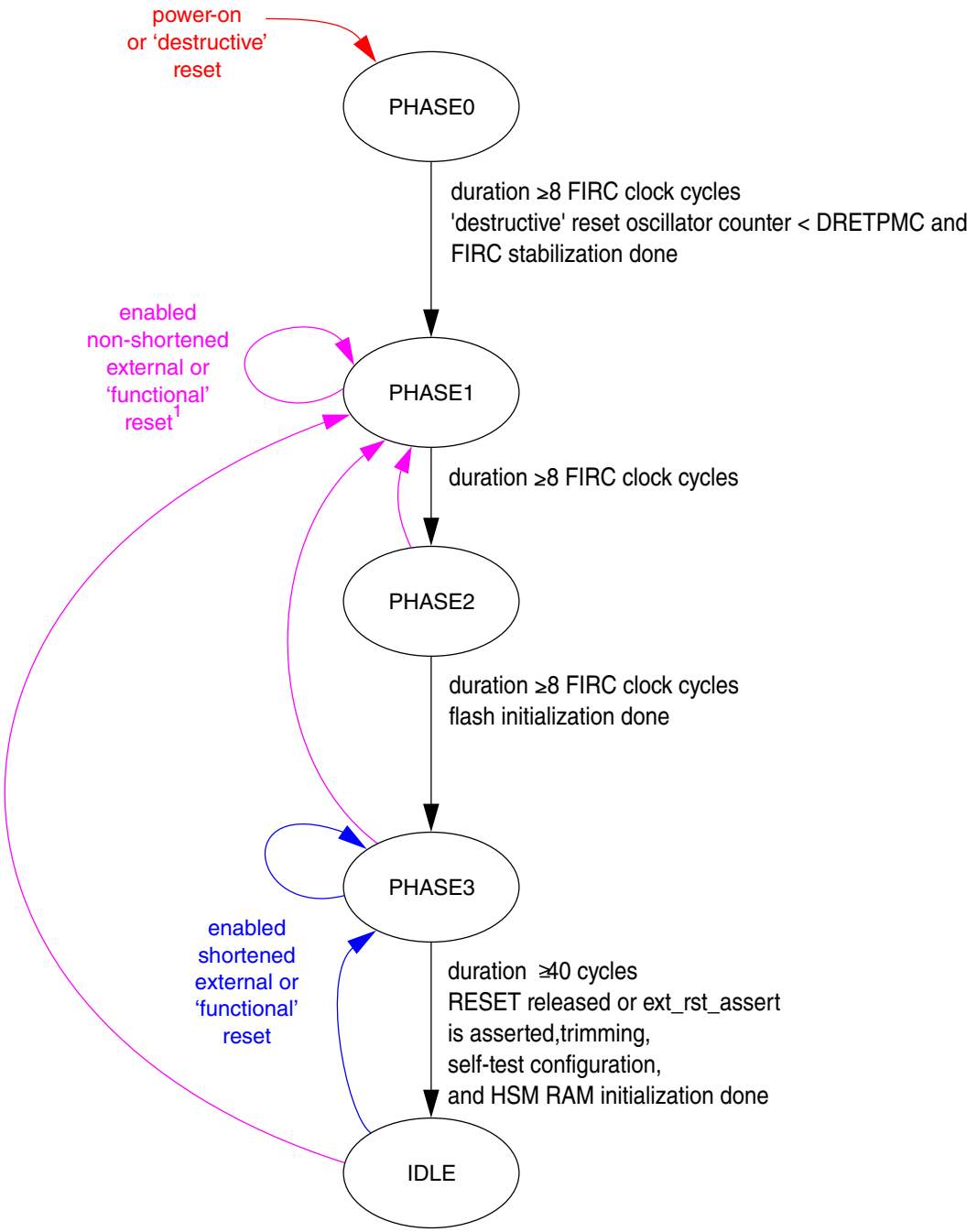


Figure 53-2. MC_RGM State Machine

53.4.1.1 PHASE0 Phase

This phase is entered immediately from any phase on a power-on or enabled 'destructive' reset event. The reset state machine exits PHASE0 and enters PHASE1 on verification of the following:

- All enabled 'destructive' resets have been processed

- All processes that need to be done in PHASE0 are completed
    - PMC and FIRC (16 MHz internal RC oscillator) stabilization
- A minimum of eight FIRC clock cycles have elapsed since power-up completion and the last enabled 'destructive' reset event
- The ‘destructive’ reset escalator counter has not reached the value in the DRET field of the RGM_DRET register

53.4.1.2 PHASE1 Phase

This phase is entered either on exit from PHASE0 or immediately from PHASE2, PHASE3, or IDLE on a non-masked external or 'functional' reset event if it has not been configured to trigger a 'short' sequence. The reset state machine exits PHASE1 and enters PHASE2 on verification of the following:

- All enabled, non-shortened 'functional' resets have been processed
- A minimum of 8 FIRC clock cycles have elapsed since the last enabled external or non-shortened 'functional' reset event

53.4.1.3 PHASE2 Phase

This phase is entered on exit from PHASE1. The reset state machine exits PHASE2 and enters PHASE3 on verification of the following:

- all processes that need to be done in PHASE2 are completed
    - flash initialization
- a minimum of 8 FIRC clock cycles have elapsed since entering PHASE2

53.4.1.4 PHASE3 Phase

This phase is a entered either on exit from PHASE2 or immediately from IDLE on an enabled, shortened 'functional' reset event. The reset state machine exits PHASE3 and enters IDLE on verification of the following:

- All processes that need to be done in PHASE3 are completed

- Trimming, start-up self test configuration, and HSM RAM initialization

- A minimum of forty FIRC clock cycles have elapsed since the last enabled, shortened 'functional' reset event
- RESET is not asserted, except if ext_rst_assert is asserted

53.4.1.5 IDLE Phase

This is the final phase and is entered on exit from PHASE3. When this phase is reached, the MC_RGM releases control of the system to the platform and waits for new reset events that can trigger a reset sequence.

53.4.2 'Destructive' Resets

A 'destructive' reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed.

The status flag associated with a given 'destructive' reset event (RGM_DES.F_<destructive reset> bit) is set when the 'destructive' reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software's responsibility to determine which reset source is the most critical for the application.

The chip’s low-voltage detector threshold ensures that when the reset corresponding to the core supply low-voltage detect is enabled, the supply is sufficient to have the ‘destructive’ event correctly propagated through the digital logic. Therefore, if a given ‘destructive’ reset is enabled, the MC_RGM ensures that the associated reset event will be correctly triggered to the full system. However, if the given ‘destructive’ reset is disabled and the voltage goes below the digital functional threshold, functionality can no longer be ensured, and the reset may or may not be asserted.

An enabled 'destructive’ reset will trigger a reset sequence starting from the beginning of PHASE0.

53.4.3 External Reset

The MC_RGM manages the external reset coming from RESET. The detection of a falling edge on RESET will start the reset sequence from the beginning of PHASE1.

The status flag associated with the external reset falling edge event (RGM_FES.F_EXR bit) is set when the external reset is asserted and the power-on reset is not asserted.

An external reset will normally trigger a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by the external reset. When RGM_FESS.SS_EXR is set, the external reset will trigger a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially when an external reset should not reset the flash.

The MC_RGM may also assert the external reset if the reset sequence was triggered by one of the following:

- a power-on reset
- a 'destructive' reset event configured via the RGM_DBRE register to assert the external reset
- an external reset event
- a ‘functional’ reset event configured via the RGM_FBRE register to assert the external reset

In this case, RESET is asserted until all conditions for exiting PHASE3 have been met with the exception of the RESET assertion check.

In addition, the MC_RGM asserts RESET when ext_rst_assert is asserted.

53.4.4 'Functional' Resets

A 'functional' reset indicates that an event has occurred after which it can be guaranteed that critical register and memory content is still intact.

The status flag associated with a given 'functional' reset event (RGM_FES.F_<functional reset> bit) is set when the 'functional' reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously, and it is software's responsibility to determine which reset source is the most critical for the application.

A given 'functional' reset can be optionally disabled by software writing bit RGM_FERD.D_<functional reset>.

Note

The RGM_FERD register can be written only once between two power-on reset events.

An enabled 'functional' reset will normally trigger a reset sequence starting from the beginning of PHASE1. Nevertheless, the RGM_FESS register enables the further configuring of the reset sequence triggered by a 'functional' reset. When RGM_FESS.SS_<functional reset> is set, the associated 'functional' reset will trigger a reset sequence starting directly from the beginning of PHASE3, skipping PHASE1 and PHASE2. This can be useful especially in case a 'functional' reset should not reset the flash module.

See the MC_ME chapter for details on the STANDBY0 and DRUN modes.

53.4.5 Alternate Event Generation

The MC_RGM provides alternative events to be generated on reset source assertion. When a reset source is asserted, the MC_RGM normally enters the reset sequence. Alternatively, it is possible for some reset source events to be converted from a reset to either a SAFE mode request issued to the MC_ME or to an interrupt request issued to the core.

Alternate event selection for a given reset source is made via the RGM_FERD and RGM_FEAR registers as shown in the following table.

Table 53-2. MC_RGM Alternate Event Selection

RGM_FERD Bit Value	RGM_FEAR Bit Value	Generated Event
0	X	reset
1	0	SAFE mode request
1	1	interrupt request

The alternate event is cleared by deasserting the source of the request (i.e., at the reset source that caused the alternate request) and also clearing the appropriate RGM_FES status bit.

Note

SAFE mode requests are generated regardless of whether the system clock is running. Interrupt requests are gnerated with the system clock, and therefore, if the system clock was disabled at the time of the event, the interrupt request isn't asserted until the system clock is re-enabled.

Note

If a masked 'destructive' reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE0, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

Note

If a masked 'functional' reset event which is configured to generate a SAFE mode/interrupt request occurs during PHASE1, it is ignored, and the MC_RGM will not send any safe mode/interrupt request to the MC_ME.

53.4.6 'Functional' Reset Escalation

'Functional' reset escalation can be used to generate a 'destructive' reset if a number of 'functional' or external resets has occurred between software writes to the RGM_FRET register. This function is enabled by writing a non-zero value to the FRET field of this register.

Once 'functional' reset escalation has been enabled, the MC_RGM increases a counter on each 'functional' or external reset which causes a reset sequence to be initiated (i.e., entrance into PHASE1 or PHASE3 from the IDLE phase) if the func_rst_escalation output is connected to one of the dest_rst inputs. This counter is cleared on a write of any value to the RGM_FRET register and on any power-on or 'destructive' reset. If the counter reaches the value in the RGM_FRET register's FRET field, the MC_RGM starts a reset sequence at PHASE0 and asserts the F_EDR bit in the RGM_DES register.

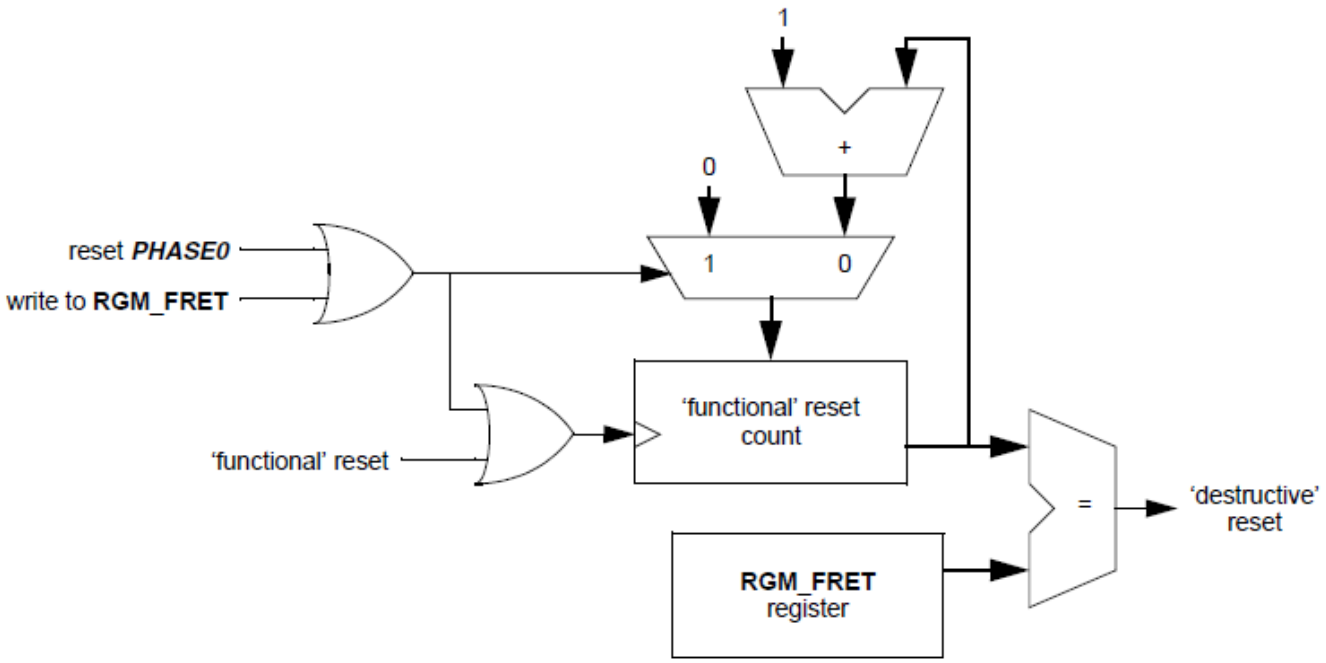


Figure 53-3. MC_RGM 'Functional' Reset Escalation to 'Destructive' Reset

53.4.7 'Destructive' Reset Escalation

'Destructive' reset escalation can be used to keep the chip in the reset state until the power-on triggers a reset sequence if a number of 'destructive' resets has occurred between software writes to the RGM_DRET register. This function is enabled by writing a non-zero value to the DRET field of this register.

Once 'destructive' reset escalation has been enabled, the MC_RGM increases a counter on each 'destructive' reset which causes a reset sequence to be initiated (i.e., entrance into PHASE0 from the IDLE phase) or an ongoing reset sequence to restart (i.e., entrance into PHASE0 from PHASE1, PHASE2, or PHASE3). This counter is cleared on a write of any value to the RGM_DRET register and on any power-on reset. If the counter reaches the value in the RGM_DRET register's DRET field, the MC_RGM enters reset PHASE0 and stays there until the next power-on reset occurs.

On a power-on reset, the dest_rst_escalation output is deasserted. It is asserted by the MC_RGM when the counter reaches the value in the RGM_DRET register's DRET field. It is not deasserted until the next power-on reset.

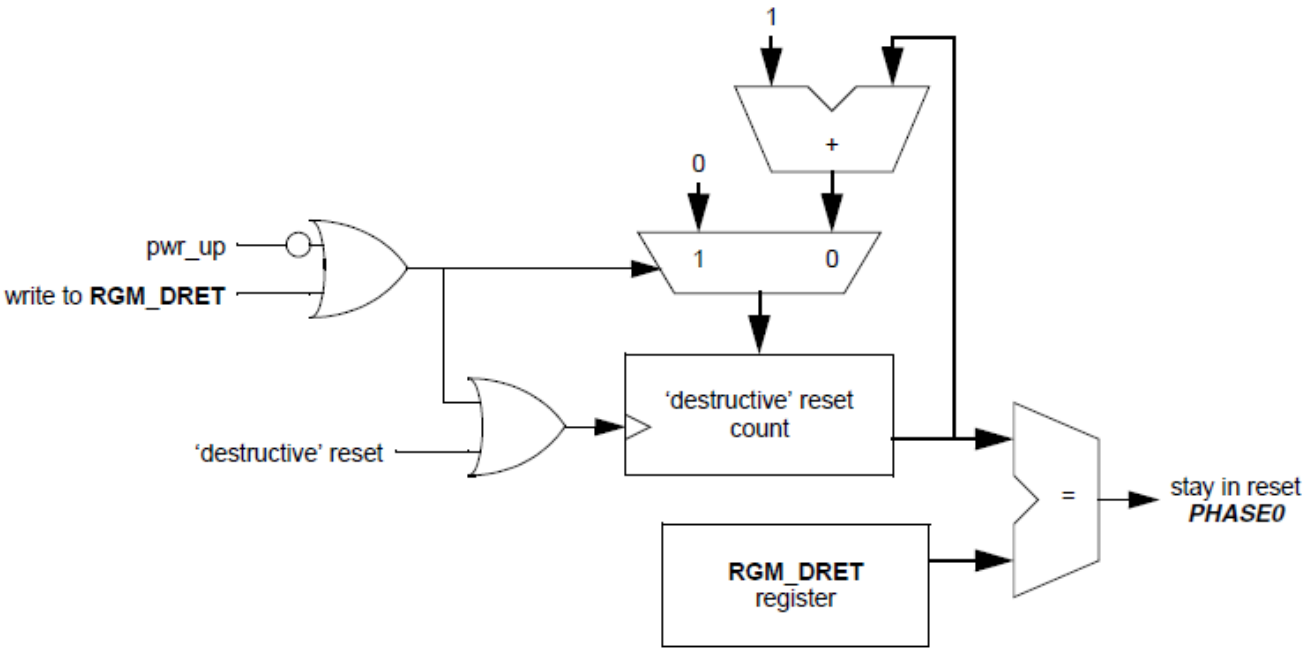


Figure 53-4. MC_RGM 'Destructive ' Reset Escalation to Stay in PHASE0

Chapter 54
System Status and Configuration Module (SSCM)

54.1 Chip-specific SSCM information

54.1.1 System Status and Configuration Module (SSCM) configuration

This section summarizes the System Status and Configuration Module (SSCM) configuration. There is one instance of the SSCM module on the MCU.

The SSCM reads the DCF records during boot and then automatically switch to Booting Core to BAF.

The device specific reset value for the SSCM_MEMCONFIG register are shown in the table Table 54-1.

Table 54-1. SSCM_MEMCONFIG reset value

Field	Field Definition	Value
JPIN	JTAG Part ID	084h
MREV	Minor Revision	0000b

The device specific reset value for the SSCM_UOPS register are shown in the table Table 54-2.

Table 54-2. SSCM_UOPS reset value

Register	Reset Value
SSCM_UOPS	0000_ 0600

Table 54-3 shows the address locations that the SSCM uses to search for startup information.

Table 54-3. SSCM startup information addresses

Description	Address
The location at which the SSCM will look for the UTEST records (UTEST_OFFSET).	0040_0300h
Location of the Boot Assist code in flash memory (BAF)	0040_4000h

NOTE

SSCM_STATUS[NXEN] will be set if any of the cores has Nexus enabled.

54.2 Introduction

54.2.1 Overview

The System Status and Configuration Module (SSCM) is pictured in the figure below.

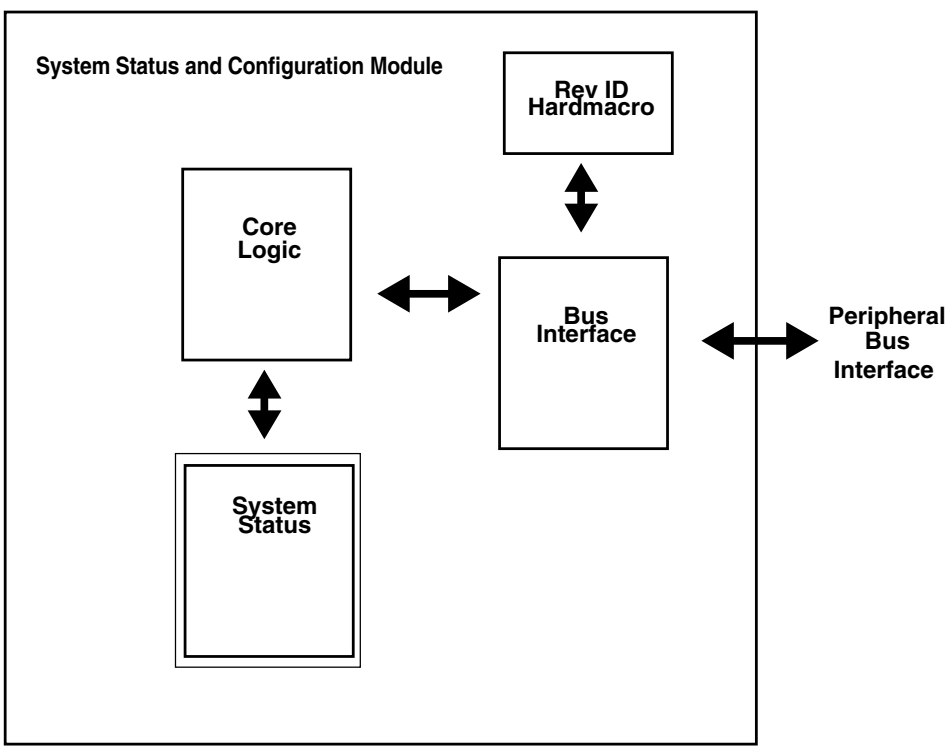


Figure 54-1. SSCM block diagram

54.2.2 Glossary

The table below shows a glossary of terms used through out the text.

Table 54-4. Glossary

Term	Definition
0011b	Binary numbers
0Fh	Hexadecimal numbers
Pin	External physical connection.
Signal	Electronic construct whose state or change in state conveys information.
X	In certain contexts, such as a signal encoding, this indicates a don't care. For example, if a field is binary coded X001b, the state of the first bit is a don't care.
y	y is used for a place holder of 1 hex digit for unknown values.

54.2.3 Features

The SSCM includes these distinct features:

- System Configuration and Status
  - Device Mode and System Status
  - Determine HSM boot vector
  - Decodes the Device Life Cycle
- Debug Status Port enable and selection
- Bus abort enable/disable

54.2.4 Modes of operation

The SSCM operates identically in all system modes.

54.3 External signal description

The SSCM provides a set of pins for the debug port (see Debug Status Port Register (DEBUGPORT) for details).

54.4 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the SSCM.

The table below shows the memory map for the SSCM. Note that all addresses are offsets; the absolute address may be calculated by adding the specified offset to the base address of the SSCM.

NOTE

All registers are accessible via 8-bit, 16-bit or 32-bit reads and/or writes. However, 16-bit accesses must be aligned to 16-bit boundaries, and 32-bit accesses must be aligned to 32-bit boundaries. As an example, the STATUS register is accessible by a 16-bit read or write, SSCM Base + 02h, but performing a 16-bit access to SSCM Base + 0003h is illegal (that is, you should not make any transfer that is not aligned to burst size boundary).

NOTE

SSCM does not generate transfer errors in all memory holes.

NOTE

Reserved IPS register access is prohibited.

SSCM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	SSCM System Status (SSCM_STATUS)	16	R	See section	54.4.1/2223
2	SSCM System Memory and ID Register (SSCM_MEMCONFIG)	16	R	See section	54.4.2/2224
8	SSCM Debug Status Port Register (SSCM_DEBUGPORT)	16	R/W	0000h	54.4.3/2225
20	SSCM HSM and User Option Status Register (SSCM_UOPS)	32	R	See section	54.4.4/2226
28	Processor Start Address Register (SSCM_PSA)	32	R	See section	54.4.5/2227
30	SSCM HSM Start Address Register (SSCM_HSA)	32	R	See section	54.4.6/2227
34	Life Cycle Status Register (SSCM_LCSTAT)	32	R	See section	54.4.7/2228

54.4.1 SSCM System Status (SSCM_STATUS)

The System Status register reflects the current state of the system.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7
Read	0	CER	0	0	NXEN	1	Reserved	0
Write		w1c						
Reset	0	0	0	0	*	1	*	0

Bit	8	9	10	11	12	13	14	15
Read	BMODE			1	0	0	0	
Write								
Reset	*	*	*	1	0	0	0	0

* Notes:
- BMODE field: Reset value depends on the device status after leaving reset.
- Reserved field: This field can reset to 0 or 1.
- NXEN field: Reset value depends on the device status after leaving reset.

SSCM_STATUS field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 CER	Configuration Error This field indicates that the SSCM has detected a configuration error during reset while loading initial device configuration. See the chip-specific SSCM information for details about sources of the error. If a non-permanent error is detected, this bit can be cleared by writing a 1. 0 No configuration problem detected by the SSCM 1 Device configuration is not correct
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 NXEN	Processor 0 Nexus enable status 0 Processor 0 Nexus disabled. 1 Processor 0 Nexus enabled
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
6 Reserved	This field is reserved. This field is for internal use. Users should ignore read values and should not attempt to write the field.

Table continues on the next page...

SSCM_STATUS field descriptions (continued)

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–10 BMODE	Device Boot Mode This field is only updated during reset. 101 Boot from BootFlash
11 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

54.4.2 SSCM System Memory and ID Register (SSCM_MEMCONFIG)

The System Memory and ID register is a read-only register which reflects the memory configuration of the system. It also contains the JTAG part ID.

Address: 0h base + 2h offset = 2h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read	JPIN										1	MREV				0
Write																
Reset	*	*	*	*	*	*	*	*	*	*	1	*	*	*	*	0

* Notes:
- MREV field: Reset value can be found in the SSCM Configuration section.
- JPIN field: Reset value reflects the JTAG part ID of the device.

SSCM_MEMCONFIG field descriptions

Field	Description
0–9 JPIN	JTAG Part ID Number **NOTE:** Reset value reflects the JTAG part ID of the device.
10 Reserved	This field is reserved. This read-only field is reserved and always has the value 1.
11–14 MREV	Minor Mask Revision
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

54.4.3 SSCM Debug Status Port Register (SSCM_DEBUGPORT)

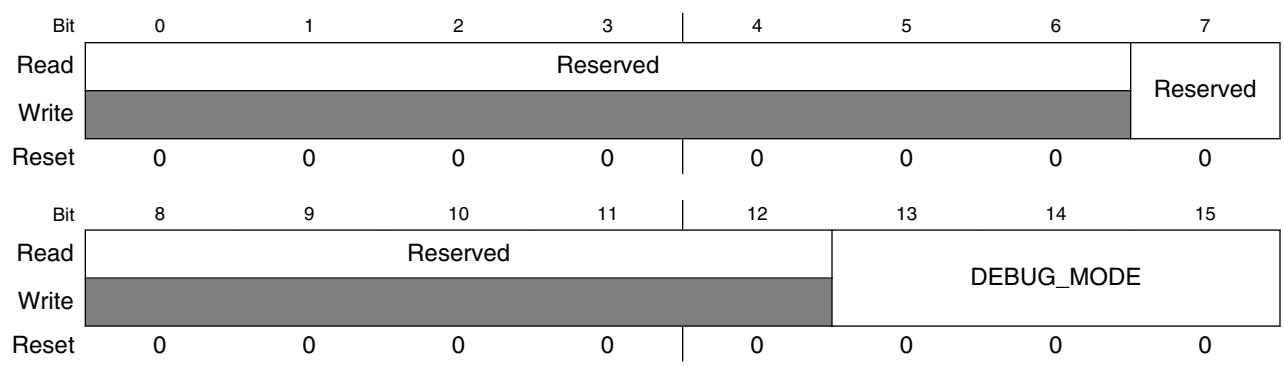
The Debug Status Port register is used to (optionally) provide debug data on a set of pins.

Table 54-5. Debug status port modes

Pin [1]	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
0	STATUS[0]	STATUS[8]	MEMCONFIG[0]	MEMCONFIG[8]	Reserved	Reserved	Reserved
1	STATUS[1]	STATUS[9]	MEMCONFIG[1]	MEMCONFIG[9]	Reserved	Reserved	Reserved
2	STATUS[2]	STATUS[10]	MEMCONFIG[2]	MEMCONFIG[10]	Reserved	Reserved	Reserved
3	STATUS[3]	STATUS[11]	MEMCONFIG[3]	MEMCONFIG[11]	Reserved	Reserved	Reserved
4	STATUS[4]	STATUS[12]	MEMCONFIG[4]	MEMCONFIG[12]	Reserved	Reserved	Reserved
5	STATUS[5]	STATUS[13]	MEMCONFIG[5]	MEMCONFIG[13]	Reserved	Reserved	Reserved
6	STATUS[6]	STATUS[14]	MEMCONFIG[6]	MEMCONFIG[14]	Reserved	Reserved	Reserved
7	STATUS[7]	STATUS[15]	MEMCONFIG[7]	MEMCONFIG[15]	Reserved	Reserved	Reserved

1. All signals are active high, unless otherwise noted

Address: 0h base + 8h offset = 8h



SSCM_DEBUGPORT field descriptions

Field	Description
0–6 Reserved	This field is reserved.
7 Reserved	This field is reserved.
8–12 Reserved	This field is reserved.

Table continues on the next page...

SSCM_DEBUGPORT field descriptions (continued)

Field	Description
13–15 DEBUG_MODE	Debug Status Port Mode. This field selects the alternate debug functionality for the Debug Status Port Table 54-5 describes the functionality of the Debug Status Port in each mode. 000 Undefined 001 Mode 1 selected 010 Mode 2 selected 011 Mode 3 selected 100 Mode 4 selected 101 Mode 5 selected 110 Mode 6 selected 111 Mode 7 selected

54.4.4 SSCM HSM and User Option Status Register (SSCM_UOPS)

Address: 0h base + 20h offset = 20h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0												HSB			HSE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	*	*	*	*

* Notes:
- HSE field: Values are determined by DCF record and life cycle - default is 0.
- HSB field: Values are determined by DCF record and life cycle - default is 0.

SSCM_UOPS field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–30 HSB	HSM Boot Configuration These bits can be set via DCF record (see HSM documentation). All combinations not listed below are reserved. 000 When the device is in BAF mode, BAF will not wait for HSM ready flag before allowing application code to start. 001 When the device is in BAF mode, BAF waits for HSM ready flag before allowing application code to start.

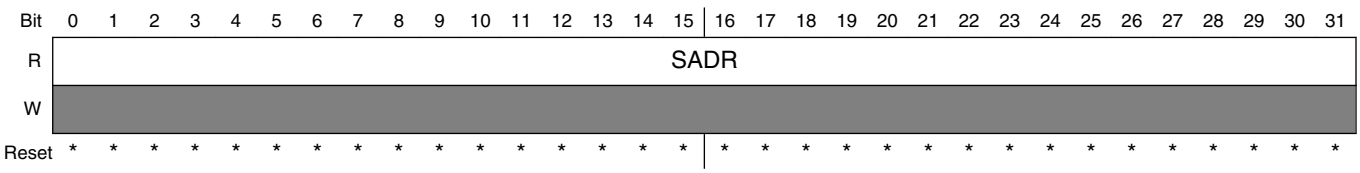
Table continues on the next page...

SSCM_UOPS field descriptions (continued)

Field	Description
31 HSE	HSM Enabled HSE is set to 1 if HSM is enabled via DCF record

54.4.5 Processor Start Address Register (SSCM_PSA)

Address: 0h base + 28h offset = 28h



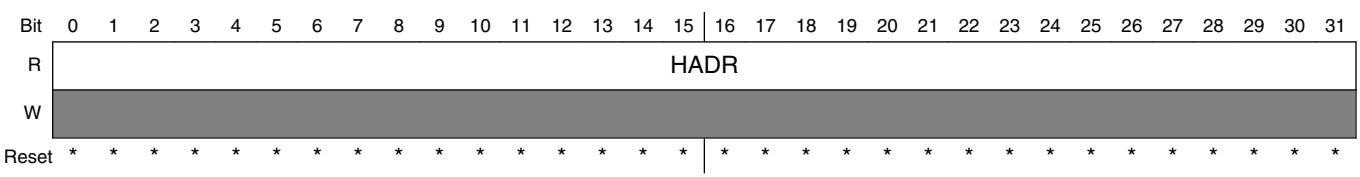
* Notes:
- SADR field: Reset value is the start address

SSCM_PSA field descriptions

Field	Description
0–31 SADR	Processor Start Address The boot processor will start executing application code from this address. In BAF mode this will show the start address of the BAF code.

54.4.6 SSCM HSM Start Address Register (SSCM_HSA)

Address: 0h base + 30h offset = 30h



* Notes:
- HADR field: HSM Start Address

SSCM_HSA field descriptions

Field	Description
0–31 HADR	HSM Start Address HSM will start executing application code from this address. The value of this register is determined by the HSM boot header search.

54.4.7 Life Cycle Status Register (SSCM_LCSTAT)

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						Reserved		0					LC		
W																
Reset	0	0	0	0	0	0	*	*	0	0	0	0	0	*	*	*

* Notes:
- LC field: Reset value is the Life Cycle of the device, and is unaffected by reset.
- Reserved field: Reset Value is '11' for a valid secured part.

SSCM_LCSTAT field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 Reserved	This field is reserved.
24–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 LC	Life Cycle Determined by the SSCM by reading the LC slots. The encoding is: 000 Failure Analysis 001 Reserved 010 OEM Production 011 Customer Delivery 100 Reserved 101 Reserved 110 MCU Production 111 In Field

54.5 Functional description

The primary purpose of the SSCM is to provide information about the current state and configuration of the system that may be useful for configuring application software and for debug of the system.

54.5.1 DCF mechanism

The DCF mechanism has been developed to handle settings of device parameters via OTP flash memory.

Starting at location UTEST_OFFSET the user can store a series of DCF records - the device will process these records during the system reset sequence before the CPU leaves reset. The first record needs to be a start record followed by an application-specific number of data records as required, and finally a stop record.

Each record is 64 bits in length. Table 54-6 shows the DCF start record. The start record must be placed at UTEST_OFFSET to indicate to the device that the following data records should be processed.

Table 54-6. DCF start record

0:31
05AA 55AFh
0:31
0000 0000h

The data records are structured similar to a CPU write instruction - 15 bits of client select, 15 bits of address, 32 bits of payload data plus a STOP bit - See Table 54-7. In a data record the STOP bit must always be programmed to 0. Some clients support a parity bit.

Table 54-7. Format of a DCF data record

0:31			
WDATA[31:0]			
0:14	15:29	30	31
CS[14:0]	ADDR[16:2]	PRTY	STOP

The stop record indicates that processing should stop. The general format of the stop record is shown in Table 54-8. Only the STOP bit needs to be 1 in order to form a stop record – All other bits are ignored. An unprogrammed location in UTEST flash will be interpreted as a stop record.

Table 54-8. DCF stop record

0:30	31
rsvd	1

Table continues on the next page...

Table 54-8. DCF stop record (continued)

0:31
rsvd

If *n* data records are to be stored in UTEST, the data structure must be as shown in Table 54-9.

Table 54-9. Series of DCF records in UTEST

ADDR offset	DATA			
00h	05AA 55AFh			
04h	0000 0000h			STOP=0
08h	WDATA[31:0]			
0Ch	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
10h	WDATA[31:0]			
14h	CS[14:0]	ADDR[16:2]	PRTY	STOP=0
…	…			
8n - 1 + 0h	reserved			
8n - 1 + 4h	reserved			1
8n + 0h				
8n + 4h				

There must never be an unprogrammed record in the data structure, as that would be interpreted as a stop record, so subsequent records would be ignored. This allows programming the records in several sessions, each time appending new records at the end of the list, as shown in Figure 54-2.

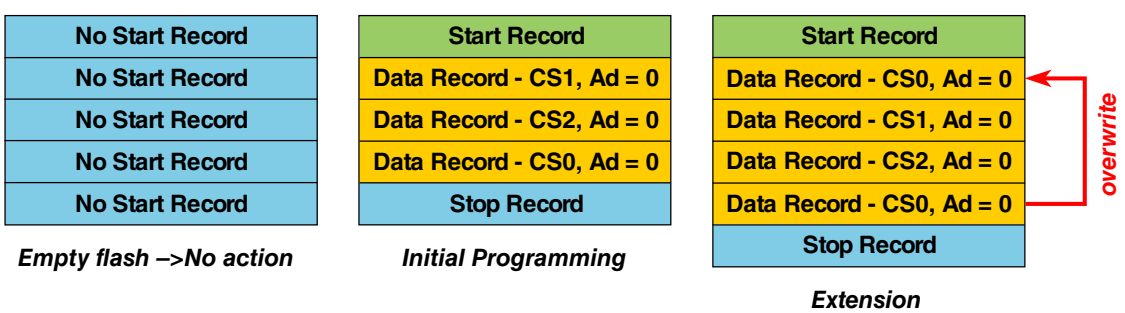


Figure 54-2. Appending DCF records

In this case, a record may overwrite a client's value which was already set by a previous record. However whether such an operation is allowed depends on the rules given for the client - some clients are write-once, only allow setting bits but not clearing, and so on.

The parity bit (PRTY) is required for some DCF records. The parity scheme used is even parity, so the number of 1s in the WDATA and PRTY fields needs to be even. (For example, if the WDATA field has the value 0000_0001h then the PRTY field needs to be set to 1, so that the total number of 1s is even.)

NOTE

When programming DCF records, ensure that programming is not interrupted and can complete without error.

54.5.2 Boot mode functionality

The device supports the following boot modes for the main boot core:

- Boot Assisted by Flash (BAF) - The device will boot from UTest NVM Flash Block, (interface with the HSM if available) and perform device setup.

If booting is not possible with the selected configuration (for example, no Boot ID is found in the selected boot location) then the device will enter static mode.

If the HSM is enabled via DCF record, the HSM will boot from the location found via the HSM boot header search. HSM boot header search.

54.5.3 BAF configuration

By default the device will boot from BAF.

In BAF mode (SSCM_STATUS[BMODE]=101b), the boot core will start executing BAF code at the location shown in the configuration section of the “Boot Assist Flash (BAF)” chapter.

54.5.4 HSM boot header search

The SSCM searches the code flash memory of the HSM module for a valid boot header (see the SSCM section of the “Device Configuration” chapter for address(es)) and provides the boot vector to the CPU of the HSM module.

The HSM Boot Header has the format as described in the table below.

Table 54-10. Boot header structure

00h	Boot Header ID	
08h	Reserved	Start Address
10h	Reserved for Configuration Bits	

A valid Boot Header ID has the value FFFF_0000_FFFF_0000h.

The first valid Boot Header found is the one which is passed to the HSM CPU. In order to switch from one Boot Header to a Boot Header at a higher address, the Boot Header at the lower address must be invalidated. This can be achieved by over-programming the Boot Header ID location with 0000_0000_0000_0000h. Since this value results in the same ECC encoding it will still be possible to read the location without causing an ECC error.

54.5.5 Life Cycle

The SSCM will determine the Life Cycle (LC) of the device by reading the LC slots from the UTEST flash memory. The read operation is done during the reset phase with normal timings and it is protected by operating monitors. In addition, a set of sanity checks executed over the LC read data guarantee the integrity of the final LC value.

At the end of the reset phase, the LC can have one of the following values:

- MCU Production
- Customer Delivery
- OEM Production
- In Field
- Failure Analysis

Life cycle can only be moved in the direction shown below:



Figure 54-3. Life cycle sequence

The LC is written into 5 slots, 128 bits each, at fixed locations in the UTEST flash memory block. Each LC slot is read in one single atomic operation and it is organized into two fields:

- the valid field
- the invalid field

Depending on the possible combination of the data programmed into these fields, each LC slot can have one of the possible 4 statuses as seen in Table 54-11.

Table 54-11. LC slot status

LC slots		LC slot value
Valid field (64 bits)	Invalid field (64 bits)	
Erased	Erased	Erased
Marked	Erased	Active
Marked	Marked	Inactive
Any other values		Illegal

"Marked" in this case means that the value has been programmed with the bit pattern 55AA_50AF_55AA_50AFh. "Erased" is detected by the bit pattern FFFF_FFFF_FFFF_FFFFh. Table 54-12 shows how the slots are arranged in memory.

Table 54-12. LC slots in memory

Offset	LC slot word
00h	Valid[31:0]
04h	Valid[63:32]
08h	Invalid[31:0]
0Ch	Invalid[63:32]

The Life Cycle is determined as shown in Table 54-13

NOTE

"Erased" means the slot returns all 1s.

Table 54-13. LC slots

LC slot 0 MCU Production	LC slot 1 Customer Delivery	LC slot 2 OEM Production	LC slot 3 In Field	LC slot 4 Failure Analysis	Resulting life cycle
Active	Erased	Erased	Erased	Erased	MCU Production
Inactive	**Active**	Erased	Erased	Erased	Customer Delivery
Inactive	Inactive	**Active**	Erased	Erased	OEM Production
Inactive	Inactive	Inactive	**Active**	Erased	In Field
Inactive	Inactive	Inactive	Inactive	**Active**	Failure Analysis
Erased	Erased	Erased	Erased	Erased	System Reset
Illegal[1]	Illegal	Illegal	Illegal	Erased	In Field

1. Any value programmed other than shown in table above for Erased, Active, and Inactive would be treated as illegal.

NOTE

When programming Life Cycle, ensure that programming is not interrupted and can complete without error.

NOTE

Any other combination of values will result in a system reset.

NOTE

No one should program “ILLEGAL” even if able to do that. Device behavior is not guaranteed for these combinations.

54.6 Initialization and application information

54.6.1 Reset

The reset state of each individual bit appears in the detailed register descriptions.

54.7 Additional safety measures

54.7.1 Spurious reset protection

The SSCM implements protection against spurious module resets (for example, resets which only effect the SSCM, but not other modules) by gating the state machines status with the expected status of the MC_RGM. If a spurious reset occurs, the SSCM will not interfere with the flash memory bus or overwrite configuration registers. However, internal SSCM status will be lost.

Chapter 55
Boot Assist Flash (BAF)

55.1 Chip-specific BAF information

55.1.1 BAF configuration

55.1.2 BAF memory map

The BAF is located in a 16 KB block of flash that is mapped adjacent to the UTEST flash memory block. It is one time programmable (OTP) and is programmed during factory test. The address space and memory used by the BAF code are shown in Table 55-1.

Table 55-1. BAF memory map

BAF component	Address
BAF address range	0040_4000h – 0040_7FFFh
BAF version number	0040_4000h
BAF code entry point	0040_4100h
BAF code exit point	4007_FD80h
BAF data area and stack	4007_FD00h – 4007_FFFFh

55.1.3 LinFLEXD_0 pin configuration

Table 55-2 shows the device pin configurations for LINFlexD_0 module.

Table 55-2. LINFlexD_0 pin configuration

Pads	Reset Function	Initial Serial Boot mode	
		Function	Pad configuration
PB[0]	GPIO Input buffer off Output buffer off	LIN0TX	Output Push/Pull medium no pullup/pulldown
PB[1]	GPIO Input buffer off Output buffer off	LIN0RX	Input:- pullup hystersis

55.1.4 Search boot header and boot options

The BAF searches the flash memory for the boot header, checking the first word location at the addresses mentioned in the table below. The blocks are searched in the order shown in Table 55-3. Once a Boot Header is found, no further blocks are searched.

Table 55-3. Locations of boot headers

Search order	Block	Address
1	16KB Code Flash block	00F8_C000h[1]
2	32KB Code Flash block	00FC_0000
3	32KB Code Flash block	00FD_8000
4	64KB Code Flash block	00FE_0000
5	16KB Code Flash block	00F9_0000h
6	16KB Code Flash block	00F9_4000h
7	16KB Code Flash block	00F9_8000h
8	16KB Code Flash block	00F9_C000h
9	16KB Code Flash block	00FA_0000h
10	16KB Code Flash block	00FA_4000h
11	16KB Code Flash block	00FA_8000h

1. Location 00F8_C000h will be searched when HSM is not enabled, otherwise it will be skipped.

The first header found that contains the value 005Ah in the first halfword is valid for booting. The whole header structure is shown in Table 55-4.

Table 55-4. Boot header structure

Address Offset	Contents
00h	Boot Header Configuration (see Table 55-5)
04h	CPU2 Reset Vector
08h	Configuration Bits

Table continues on the next page...

Table 55-4. Boot header structure (continued)

Address Offset	Contents
0Ch	Configuration Bits
10h	CPU0 Reset Vector
18h	Reserved for future use

The first header found that contains the value 005Ah in the first halfword is valid for booting. The whole header structure is shown in Table 55-5

Table 55-5. Boot header configuration

Offset 00h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0				0				5				A			
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0				0				0				0	0	CPU0	CPU2

Table 55-6. Boot header configuration field description

Field	Description
CPU0	CPU0 is enabled (will start execution at CPU0 Reset Vector)
CPU2	CPU2 is enabled (will start execution at CPU2 Reset Vector)

The cores that are enabled in the boot header configuration field are enabled by programming MC_ME_CCTL*n*[DRUN] register to run in all DRUN modes and their reset vector is set in register MC_ME_CADDR*n*. Then, the memory area of the BAF is protected from execution for functional safety considerations. This is achieved by disabling BAF execution in PFLASH control register3 (PFLASHC_PFCR3[BAF_DIS]). Finally, a mode change (from DRUN to DRUN again) is requested from the MC_ME module to activate the new core configuration and make the cores jump to their respective reset vectors. At this point the BAF has finished its operation.

55.1.5 MPC5746C cores/BAF CPU*n*

This chapter uses the term CPU*n* (n=0, 1, or 2) when referring to a specific processor core. See Table 55-7 for details specific to the MPC5746C device.

Table 55-7. MPC5746C core to BAF CPU*n* relationships

CPU*n*	MPC5746C core
CPU0	e200z420 (Z4)
CPU2	e200z210 (Z2)

55.1.5.1 BAF dependent reset values

The register values may be changed during execution of BAF. The table below provides typical register values for such modules.

Table 55-8. BAF execution dependent reset values

Register	Remark
MC_ME_IS	Typically: 0x1 during Flash Boot may have other faults set due to SAFE and NMI Handling
MC_ME_IMTS	Typically 0x0 during Flash Boot; may have 0x1 if BAF enters to SAFE
MC_ME_DMTS[PREVIOUS_MODE]	SAFE or DRUN
MC_ME_GS[CURENT_MODE]	SAFE or DRUN
MC_ME_CS, MC_ME_CCTLX, MC_ME_CADDRX	Undefined, Customer specific configuration
PRAMC0_PRC1[FT_DIS]	0: During Boot 1: Fast PLL STBY Exit
FCCU_NCF	Z4A Machine Check Fault when boot with NMI fault already latched not otherwise
FXOSC_CTL	I_OSC set. After Serial Boot only.
PFLASH_PFCR3	BAF_DIS = 1

55.1.6 BAF NMI event handling

BAF executes under different conditions viz. Power-On-Reset (POR), Destructive Reset or any Functional Reset (Long or Short). Due to device configuration, there can be a case where pending FCCU NMI events are latched in the device. These NMI events are masked as described in NMI/WKPU interface. When BAF issues device mode transition to transfer control to application (MC_ME_CADDR *x* [RMC] = 1), the FCCU NMI event gets unmasked causing exception until RMC take effect and resets the core. The exception to CPU0 causes an additional fault (CPU0 Machine check) to be latched in the FCCU. Reset to the core also masks FCCU NMI Reaction.

As a workaround, the application can install a CALLBACK – BAF callback DCF client to initialize FCCU fault reaction to default state (No reaction for NMI Faults). This ensures that NMI faults occurring (during the BAF execution) are logged in the system. Application, then in its boot code, can take corrective action and re-initialize FCCU.

As a best practice, application must check for any pending FCCU events and clear them.

55.2 Introduction

The MCU is booted through a collaboration of several blocks, hardware, and firmware. The first boot phases are performed by a state machine inside the System Status and Configuration Module (SSCM). Once completed, SSCM sends the reset vector to the boot core, which sets the program counter to the BAF code starting address.

One of the first tasks executed by BAF is a read to the DCF records starting address, i.e., 0040_0300h. An uncorrectable ECC error associated with the DCF records starting address indicates an invalid boot scenario. Prior to control being passed to BAF, SSCM parses DCF records after obtaining the DCF records starting address. If an uncorrectable ECC error results, the SSCM is unable to process any DCF records, including mandatory DCF records and customer DCF records, making any boot sequence executed subsequent to the ECC error invalid. If BAF code detects an uncorrectable ECC error at the DCF records starting address, it issues a destructive reset via the MC_ME module.

In addition to checking the DCF records starting address for an uncorrectable ECC error, the BAF clears any ECC errors latched in the C55FMC flash memory's MCR register during SSCM processing.

The BAF code then checks the life cycle of the device. If the life cycle is FAIL_ANALYSIS, the BAF enters a loop in which it services the watchdog. The BAF does not execute further since the read access to flash memory boot sectors for the cores may be disabled if the device life cycle is FAIL_ANALYSIS. Otherwise, it searches for a boot header and boots the application code in internal flash memory.

If no boot header is found in internal flash memory, it downloads application code serially using the LINFlexD module (see Serial boot configuration for configuration details).

55.3 BAF image header

BAF image contains a 256-byte header starting from the BAF version number (see table "BAF memory map" table in the "BAF Configuration" section of this chapter). The header contains important information such as BAF version. The BAF header is explained in Table 55-9.

Table 55-9. BAF image header

Address offset	Size (bits)	Field description
0000h	32	BAF image version
0004h – 00FFh	Reserved	

55.3.1 BAF image version

The BAF version is a 32-bit field in the image header at the BAF starting address (see the "BAF memory map" table in the "BAF Configuration" section of this chapter). The 32-bit field is explained in Table 55-10.

Table 55-10. BAF image version

Field	Description
[0:7]	Major Number: This field contains the major version number for the BAF image
[8:15]	Minor Number: This field contains the minor version number for the BAF image
[16:23]	BAF Type: This field contains customer specific information.
[24:31]	Reserved

55.4 Functional description

A functional description of the BAF modes is contained in the following sections.

55.4.1 Boot modes

Depending on the state and the content of its flash memory the device can enter one of four different boot modes:

- Boot from internal flash – If the internal flash contains application code with a valid boot header, the application is booted. The boot header contains flags that enable/ disable individual cores and their reset vectors.

- Serial boot – If no valid boot header is found and the current life cycle phase of the device is CUST_DEL (Customer Delivery) or MCU Production, an attempt is made to download the application by a serial protocol over LINFlexD physical interface. The downloaded code is then executed by the initial boot core.
- Static mode – The device enters power safe mode and waits for an external reset.
- Standby Exit Mode – The BAF optionally supports booting when the chip exits standby mode. This mode is entered when the user programs the BAF start address in the Mode Entry Core Address register (MC_ME_CADDR*x*). This mode is supported when software programs only one core out of standby exit and its corresponding MC_ME_CADDR*x* register is programmed with the BAF Entry Point (see table 'BAF memory map' at the beginning of this chapter). If multiple cores are programmed to execute at standby exit, and one or multiple cores are programmed to execute BAF, then the BAF treats this as an error condition and places the chip in safe mode (MC_ME_SAFE_MC) and wait for event.

55.4.2 Device initialization

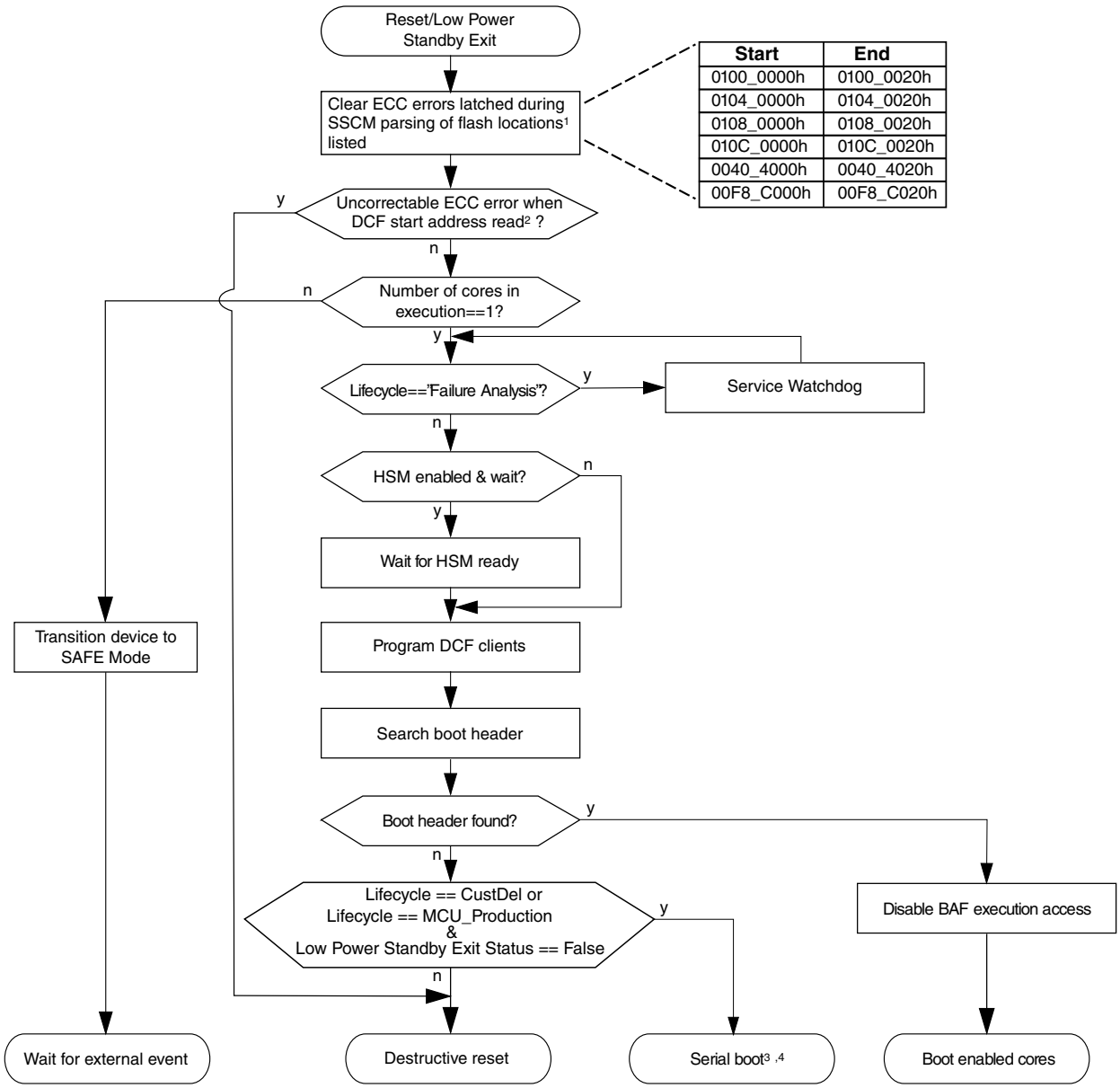
The BAF performs a minimal initialization of core registers and clocks required to enter the above described boot modes, then attempts to interact and the Hardware Security Module (HSM) in the following way:

1. BAF checks for multiple cores in execution. If more than one core is executing, the device is put in SAFE mode and executes a wait instruction in a loop.
2. HSM: If the HSM is enabled and the "wait for HSM" flag is present, then the boot core waits for the HSM to become ready. It optionally executes a random instruction sequence to create random noise on the instruction bus while waiting.
3. BAF sets up the exception vector table to catch machine check exceptions, which may occur if there are any uncorrectable ECC errors in the BAF execution path. In the event of an uncorrectable ECC error, the BAF issues a destructive reset from the machine check exception handler if the exception is due to the reading of instructions from the BAF region. If an exception occurs during reading of the Boot header, the BAF ignores the ECC error and jumps to next boot header. If the machine check exception occurs due to any other reason during BAF execution, the BAF will still issue a destructive reset. BAF programs the Mode Entry module’s MC_ME_MCTL[TARGET_MODE] to issue a destructive reset.
4. BAF uses memory of the system RAM as in the 'BAF memory map' table (at the beginning of this chapter) for its stack, data and any code that executes from volatile memory. During the BAF start-up procedure, it initializes this section of memory so that there is no ECC error when BAF accesses this memory area.

5. BAF enables instruction cache and branch prediction when executing on Z4 cores. When executing on Z2 core (out of Standby), BAF enables Branch prediction. Both Cache and Branch prediction are disabled before releasing the control to the application.
6. In case of standby exit, BAF checks the source of system clock. If the system clock is sourced from, PLL then RAM wait state is set to 1.

55.4.3 Flow of control

Figure 55-1 shows the overall flow of control in the BAF. The individual steps are explained in the following sections.



1) These locations are scanned by SSCM for its internal operations.
2) DCF records start address can be either the start of DCF records in UTEST flash or the address specified by the Soft DCF Record Start Address. See "Soft DCF clients" section.
3) Serial boot is not available on all devices. Please check the product reference manual for details.
4) SWT0 is disabled after serial boot regardless of the configuration specified in the UTEST_MISC[SWT0_WDG_DIS] register field.

Figure 55-1. Flow of control

55.4.4 Optionally remap calibration RAM

BAF supports remapping of Calibration RAM through Soft DCF Records.

55.4.5 Optionally wait for HSM

If the HSM is enabled (SSCM_UOPS[HSE] = 1), the boot core will need to wait for the HSM to become ready (if SSCM_UOPS[HSB = 1]). The core then polls the HSM until a ready flag is set. Both SSCM registers can be set by a DCF record. The boot core polls the HSM to HOST FLAGS register (HSM_HSM2HTF) until its least significant bit is set. The boot core also services the watchdog while polling the HSM_HSM2HTF register. This is just in case the HSM is not ready in 16 ms, servicing the watchdog prevents the system going to reset.

55.4.6 BAF SAFE mode event handling

BAF executes under different conditions viz. Power-On-Reset (POR), Destructive Reset or any Functional Reset (Long or Short).

Due to device configuration, the device mode can be SAFE or DRUN during BAF execution. BAF issues a DRUN Mode transition to enable any peripheral or transfer control to application. If the device mode is SAFE or changes to SAFE during ongoing DRUN mode transition, then BAF issues an additional SAFE mode transition. After BAF execution when the control is transferred to the application, the device mode can be either SAFE or DRUN. It is recommended that the application (in its bootstrap) should take necessary action after idenitifying the current mode.

NOTE

Safe mode handling is only available in Flash Boot use case. Serial boot mode does not support SAFE mode handling as XOSC is disabled in SAFE mode.

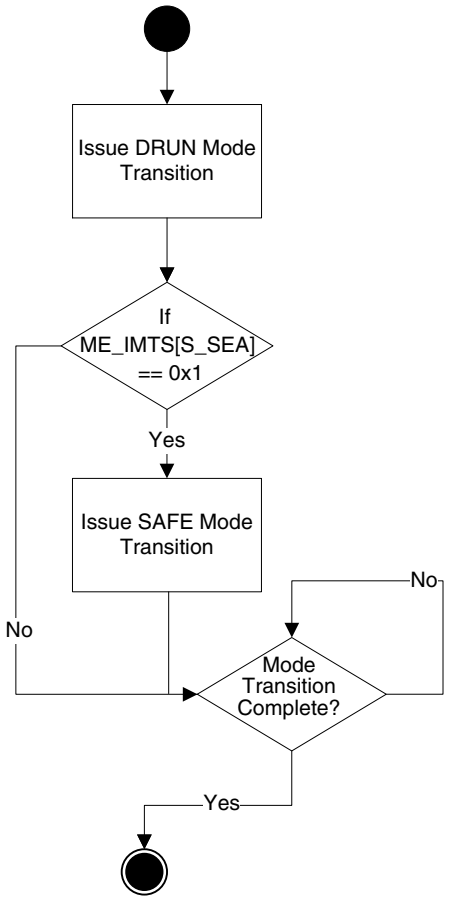


Figure 55-2. SAFE Mode Handling

55.4.7 Initialization of BAF DCF clients

55.4.7.1 Soft DCF clients

DCF clients are 32-bit hardware registers that are written by the SSCM during reset with data that is stored in DCF record format in UTEST region of flash memory. These DCF records are used to configure module registers, and select device operating modes.

The BAF extends the DCF record concept by implementing soft DCF clients. The 64-bit DCF records include:

- 32-bits of data
- 15-bit chip select field (selects the module which implements the DCF clients)
- 15-bit address field (selects a specific DCF client within the selected module)
- Parity bit (not used with UTEST DCF records)
- Stop bit (to indicate the end of DCF record has been reached)

As part of its execution flow, BAF searches through the list of DCF records in UTEST flash to determine if any of the records are to be processed by BAF.

The search starts at one of two addresses: either the start address of DCF records in UTEST (DCF Start Record), or the address specified in UTEST (Soft DCF Record Start Address). If either of the two least significant bits of the 32-bit address stored at the Soft DCF Record Start Address is set, the default start address at the DCF Start Record is used. If the two least significant bits are clear, this 32-bit value is assumed to be the start address of the search.

The default setting of the two least significant bits is 11b (erased value of flash memory is FFFF_FFFFh). A user may intentionally program a dummy address with the least two significant bits set in order to disable the search for an alternate address (as the UTEST block is OTP). Valid search start address must start on a 4-byte boundary.

The DCF record search finishes when a DCF record is read with a STOP bit set. The STOP bit may be set because that memory location is not programmed yet (so the list can be added to) or it can be an intentionally programmed STOP record to indicate that no more records are added to the list. BAF does not execute the DCF record with STOP bit set. BAF parses the STOP bit before parsing the rest of the records.

BAF uses Chip Select 14 (CS14) for BAF DCF clients. For records with CS14 asserted, the BAF uses the address field (address[14:0]) in DCF records to determine how to process the 32-bits of data. The soft DCF clients programmed by BAF are listed in Table 55-11.

A group of four Soft DCF clients is used to provide a mechanism to write configuration information to address locations. The first DCF client receive 32-bit address information. The next three DCF Clients receive the data that is to be written to the address contained in the first client. The three data clients allow 32-bit, 16-bit and 8-bit data to be written to the address in the first client. These four clients can be accessed repeatedly to write whole string of data to memory.

A set of two Soft DCF Clients are used to provide the functionality of remapping calibration RAM. The first DCF record is used to specify the address where the Platform Flash Calibration Remap Descriptor (PFCRD) image is stored. The second soft DCF client provides the address of the function that performs the remapping. The details can be found in section DCF client - PFCRD_IMAGE.

The serial boot DCF client provides a mechanism which can configure the BAF to perform serial boot or boot from internal flash. Details can be viewed in SER_BOOT_CBACK – BAF serial boot DCF client.

Table 55-11. BAF DCF client list

Address	Name	Label	Description
0000h – 01FFh	Reserved	—	Reserved for future use
0200h	Address	ADDR	The address to which DATA*n* is written.
0201h	32-bit data	DATA32	The 32-bit data that is written to ADDR.
0202h	16-bit data	DATA16	The 16-bit data that is written to ADDR.
0203h	8-bit data	DATA8	The 8-bit data that is written to ADDR.
0204h – 020Fh	Reserved	—	Reserved for future use.
0210h	Callback Address	CALLBACK	The address of a callback function.
0211h	Serial boot Callback address	SER_BOOT_CBACK	The address of serial boot callback function.
0212h	Platform Flash Calibration Remap Descriptor Image	PFCRD_IMAGE_ADDR	The address where Platform Flash Calibration Remap Descriptor Image is stored.
0213h	Platform Flash Remap Callback	PFCRD_REMAP_CALLBACK	The address to the Platform Flash Calibration Remap Callback function.
0214h – 7FFFh	Reserved	—	Reserved for future use.

NOTE

BAF boots with a watchdog timeout window set to 16ms. Parsing more and more DCF Records may increase boot time. But it should be taken care that number of DCF records should not be very high as it can cause watchdog window timeout.

55.4.7.2 BAF DCF client - ADDRESS

BAF parses the DCF records. If BAF finds a DCF record with CS14 set and Address field as 0200h while parsing, BAF programs this DCF client. This client is written with a 32-bit address from the Data section of DCF record. This 32-bit address is used in conjunction with any one of the DATA32, DATA16 or DATA8 DCF Data clients. BAF does not immediately use the ADDRESS information, but it is stored and used later in conjunction with a DATAX DCF client (DATA32, DATA16 or DATA8). Then BAF writes the data to the address in memory specified by this ADDRESS DCF client.

The user can create a series of data writes to memory by programming a series of DCF records consisting of pairs of records. The first DCF record of the pair writes to ADDR to provide the address. The second DCF record of the pair writes to the DATA client (DATAX) to provide the data. Subsequent pairs of DCF records can be programmed to write to more memory locations. The number of pairs of DCF records that can be programmed is only limited by the amount UTEST space set aside for DCF records.

Table 55-12. BAF DCF ADDR client configuration

DCF Client Address 0200h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ADDR[0:15]															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ADDR[16:31]															

Table 55-13. BAF DCF ADDR client description

Field	Description
ADDR[0:31]	32-Bit Address DCF client. This Address value is used in conjunction with a DATAX value. The BAF writes DATAX value to ADDRESS value in the memory space of the MCU

55.4.7.3 BAF DCF client - DATAX

CAUTION

While using DATAX DCF records, it should be taken in account that BAF does not take care of ECC scheme and can generate ECC, if write is performed in memory locations which are protected by ECC and are not pre-initialized as required.

55.4.7.3.1 DATA32 – 32-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0201h while parsing, BAF programs this 32-bit DCF client. BAF takes the memory address provided by the ADDR client and writes this 32-bit data to the address. (Only 32-bit aligned writes are supported)

Table 55-14. DATA32 DCF client configuration

DCF Client Address 0201h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DATA[0:15]															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DATA[16:31]															

Table 55-15. DATA32 DCF client description

Field	Description
DATA[0:31]	32-Bit DCF Data

55.4.7.3.2 DATA16 – 16-bit data DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field set as 0202h while parsing, BAF programs this 16-bit DCF client. BAF takes the memory address provided by the ADDR client and writes least significant 16-bit data to the address. Data in the 16 most significant bits are ignored. (Only 16-bit aligned writes are supported)

Table 55-16. DATA16 DCF client configuration

DCF Client Address 0202h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Reserved															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DATA[0:15]															

Table 55-17. DATA16 DCF client description

Field	Description
DATA[0:15]	32-Bit DCF Data

55.4.7.3.3 DATA8 – 8-bit data DCF client

The BAF parses the DCF records. During parsing, if the BAF finds a record with CS14 set and address field equal to 0203h, the BAF programs this 8-bit DCF client. The BAF takes the memory address provided by the ADDR client, and writes least significant 8-bit data to the address. Data in the 24 most significant bits are ignored.

Table 55-18. DATA8 DCF client configuration

DCF Client Address 0203h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Reserved															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Reserved								DATA[0:7]							

Table 55-19. DATA8 DCF client description

Field	Description
DATA[0:7]	8-Bit DCF Data

55.4.7.4 CALLBACK – BAF callback DCF client

BAF parses the DCF records. If BAF finds a record with CS14 set and address field as 0210h while parsing, BAF programs this callback DCF client. This client is written with a 32-bit start address of a function which is invoked by BAF. The BAF code checks the start address is within the UTEST or within the BAF flash memory blocks, and then passes program flow to this address. If the address is found to be outside the UTEST and BAF flash memory blocks, no function call is executed. The callback is aborted and BAF moves on to the next DCF records. The function that is called is expected to comply with Power.org EABI and to terminate in an orderly manner. An incorrectly written function causes unpredictable operations in BAF. No parameters are passed to this function and no return value is expected.

Warning

Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

BAF executes the callback function as the DCF records are parsed. Therefore several callback functions maybe executed by writing several DCF records that write to the callback DCF clients. The callback functions are executed in the same order as the DCF records.

55.4.7.5 SER_BOOT_CBACK – BAF serial boot DCF client

The BAF parses DCF records. If it finds a record with CS14 set and an address field of 0211h during parsing, the BAF programs this serial boot DCF client. This DCF client is written with a 32-bit start address of a function which is invoked by the BAF. The BAF code checks to see if the start address is within the UTEST or BAF flash memory blocks, then program flow is passed to this address if it is within either of these blocks. If the address is outside the the UTEST and BAF flash memory blocks, no function call is executed. The DCF client execution is aborted, and the BAF moves to the next DCF record. The function that is called is expected to comply with Power.org EABI and to terminate in an orderly manner. An incorrectly written function causes unpredictable operations in the BAF.

Warning

No parameters are passed to the function invoked by the BAF.

Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

The serial boot callback function returns a character value. The return value informs BAF if serial or normal internal flash boot should be executed (see Table 55-20 for details).

Table 55-20. Serial boot callback function return value definition

Return values	Description
00h	Boot from internal flash using standard search for boot record (see "Search boot header and boot options" at the beginning of this chapter).
01h	Serial boot using LINFlexD (in UART mode). See Optionally perform serial boot for details.
02h – FFh	Reserved

55.4.7.6 DCF client - PFCRD_IMAGE

55.4.7.6.1 PFCRD_IMAGE_ADDR – BAF Platform Flash Calibration Remap Image Address DCF Client

The BAF parses DCF records. If BAF finds a record with CS14 set and address field as 0212h, BAF programs this DCF Client. The client is written with 32-bit address from the data section of the DCF record. The 32-bit address is not used immediately but the value is stored and later used in conjunction with Platform Flash Calibration Remap Callback DCF record. Then BAF uses these user specified value to perform Platform Flash Calibration Remap.

The user can have multiple PFCRD_IMAGE_ADDR DCF clients programmed in the UTEST Area. Only the last DCF Client programmed before PFCRD_REMAP_CALLBACK will be used. The number of pairs of DCF records that can be programmed is only limited to the amount of UTEST space set aside for DCF records.

The address specified in this DCF record must lie in the UTEST region.

The format of Platform Flash Calibration Remap Descriptor Image is shown in Table 55-21.

Table 55-21. Remap configuration table

Address offset	Name	Contents
00h	Reserved	Reserved
04h	ADDR_SWID	Address of 64-bit ID Value in SW Flash
08h	DATAID	64-bit ID value of the calibration data
0Ch		

Table continues on the next page...

Table 55-21. Remap configuration table (continued)

Address offset	Name	Contents
10h	PFCRDE	CRD Enable Flags
14h	LSTARTADDR0	PFCRD0, Logical start address
18h	PSTARTADDR0	PFCRD0, Physical start address
1Ch	CRDSize0	PFCRD0, Master Enable and CRDSize0
20h	LSTARTADDR1	PFCRD1, Logical start address
24h	PSTARTADDR1	PFCRD1, Physical start address
28h	CRDSize1	PFCRD1, Master Enable and CRDSize1
…	…	…
68h	LSTARTADDR7	PFCRD7, Logical start address
6Ch	PSTARTADDR7	PFCRD7, Physical start address
70h	CRDSize7	PFCRD7, Master Enable and CRDSize7

NOTE

For definition of LSTARTADDR, PSTARTADDR and CRDSize, see Flash memory controller

Table 55-22. DCF Platform Flash Calibration Remap Descriptor Image Address

DCF Client Address 0212h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Address[0:15]															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Address[16:31]															

55.4.7.6.2 PFCRD_REMAP_CALBACK – BAF Platform Flash Calibration Remap Callback DCF Client

The parses DCF Records. If BAF finds a record with CS14 set and address field as 0213h, BAF programs this DCF Client. The client is written with 32 bit start address of a function which is invoked by the BAF. Following checks are performed by the BAF code before executing the function call:

- The start address of the function specified by this DCF client must be within the UTEST area or BAF Flash memory blocks.
- PFCRD_IMAGE_ADDR DCF record is programmed correctly before this DCF Client.

If either of the checks fails, BAF then moves forward to parse the next DCF record. The function that is called is expected to comply with Power.org EABI and to terminate in an orderly manner. An incorrectly written function may cause unpredictable operations in the BAF.

NOTE

The address to the remap configuration table specified in PFCRD_IMAGE_ADDR DCF record is passed as a parameter to the function invoked by the BAF.

Excessively long callback functions could cause the watchdog timer to time-out and cause a reset.

This callback function returns no value to the BAF.

Table 55-23. DCF Platform Flash Calibration Remap Callback Function

DCF Client Address 0213h															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Address[0:15]															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Address[16:31]															

Customer can either chose to implement its own remap function or make use of the default sample implementation is provided in BAF. The entry point to the sample function is shown in Table 55-24.

Table 55-24. PFRCD entry point

Address	Description
0040_40C0h	Platform Flash Calibration Remap Callback Function address

There is no restriction on number of times Platform Flash Calibration DCF record pair is programmed. It is only limited by the available size in the UTEST Area.

NOTE

Only the last DCF record pair programmed will be effective.

55.4.7.6.3 Implementation details of BAF supplied remap function

Input to the function is the pointer to Table 55-21. Figure 55-3 shows the flowchart of the default sample implementation.

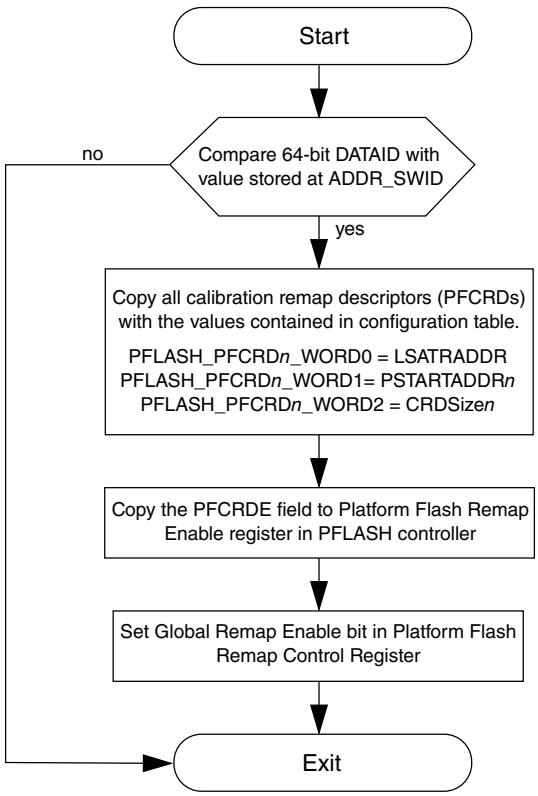


Figure 55-3. BAF supplied Calibration Remap Flowchart

55.4.8 Optionally perform serial boot

If no boot header is found in any of the locations mentioned above, the BAF determines the Life Cycle status of the device. If the Life Cycle is in CUST_DELIV (Customer Delivery) or MCU Production, it attempts a serial boot. Otherwise, the boot has failed and BAF issues a destructive reset.

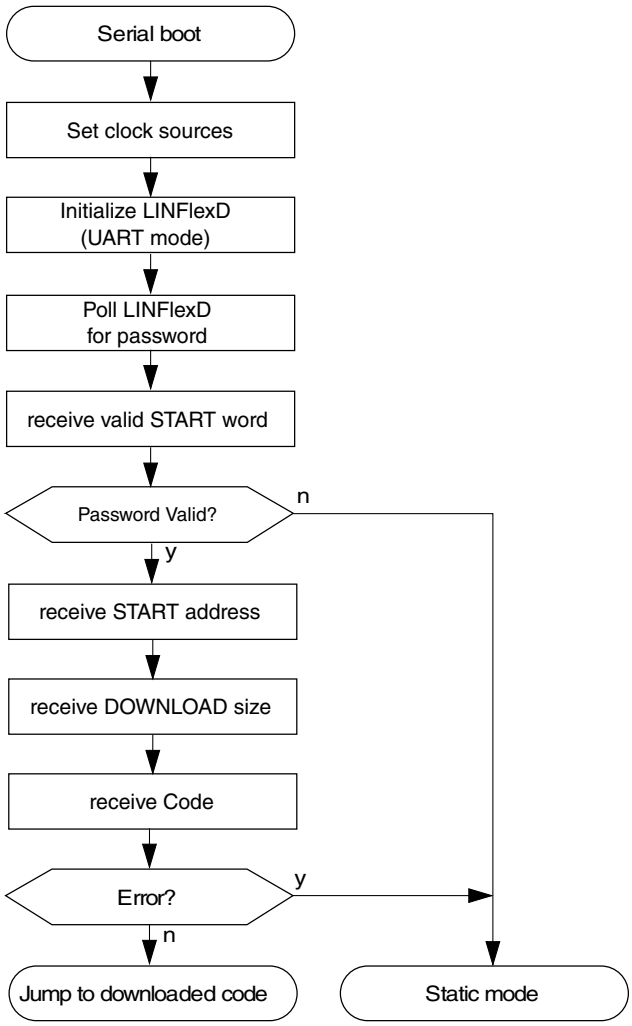


Figure 55-4. Flow of control during serial boot

The clock source in the MC_CGM module is configured and the PADs are set up for LINFlexD and signals (see Serial boot configuration for details), then the code is downloaded using LINFlexD in Serial Boot mode. In the absence of errors causing the device to be placed in Static mode, the device state is restored to resemble the out-of-reset state and the downloaded code is executed.

Serial boot allows the download of application to internal SRAM via the LINFlexD interface. After downloading the application image to SRAM, the BAF jumps to the start address provided as part of download protocol mentioned below and executes the downloaded application.

At the start of the process, LINFlexD is configured in passive listening mode. As soon as the interface receives a valid message, the receiving interface is configured in full Tx and Rx mode.

The BAF program:

- Checks if the first byte is FEh of the password. If not, continue polling.

55.4.8.1 Serial boot mode protocol

NOTE

Serial boot is not allowed from standby mode exit.

From a high-level perspective, the download protocol over LIN interface follows steps below:

1. Host transmits the 64-bit password to MCU.
2. Host transmits start address, size of downloaded code in bytes, NO_ECHO bit and VLE bit to MCU.
3. Host transmits a sequence of data bytes that are to be executed.
4. CPU0 executes code from start address provided in 2

NO_ECHO bit signifies if actual download data to be echoed or not. If NO_ECHO bit is set, then data will not be echoed otherwise if NO_ECHO bit is clear, then data will be echoed.

Each step must be complete before the next step starts.

55.4.8.2 LINFlexD download protocol

The communication is performed in half-duplex mode, Any transmission from the host is followed by the MCU transmission:

1. The host sends data to the MCU and starts waiting.
2. The MCU echoes the received 8-byte of password, 4-byte of load address and 4-byte of VLE-bit, NO_ECHO bit and 30-bit of download size to the host. The MCU echoes actual download data depending upon NO_ECHO bit.
3. The host verifies whether the echo is correct.
   - If the data is correct, the host can send the next byte of data.
   - If data is not correct, it is assumed the MCU has not correctly received the data and the only practical option is for the Host to reset the MCU and start again.

A more detailed description of these steps follows.

All multi-byte data structures are sent with MSB first.

The tool sends data in the following order:

1. 8 bytes that represent the password.
2. 4 bytes that represent the start address where the code is to be downloaded.

3. 4 bytes that represent the number of bytes to be downloaded, NO_ECHO bit and the VLE bit.
4. x bytes of code that is downloaded to SRAM and executed, (x must match the value specified in 3).

55.4.8.3 Download 64-bit password and password check

The first 64 bits received represent the password. The MCU only supports public passwords. The received password is always compared with the public password FEED_FACE_CAFE_BEEFh.

- If the correct password is received, BAF continues its task.
- If an incorrect password is supplied, BAF puts the device into static mode(low power safe mode).
- If 64 bits of password data is not received after approximately 30s, BAF causes a destructive reset of the MCU.

55.4.8.4 Download start address, VLE bit, NO_ECHO bit and code size

After the password, the next 8 bytes received by the MCU contains a 32-bit start address, the VLE mode bit, NO_ECHO bit and 30-bit code length as shown in Table 55-25.

The VLE bit (Variable Length Encoding) is used to indicate for which instruction set the downloaded code is compiled. The UART serial boot protocol supports download of VLE Code (VLE = 1) and BookE Code (VLE = 0).

The MCU only supports VLE code, i.e VLE=1.

The NO_ECHO bit is used to indicate if download data should be echoed by the MCU. If the NO_ECHO bit is set (for example, NO_ECHO = 1), then only 8 bytes of password, 4 bytes of download address, and 4 bytes of download size are echoed, but download data is not echoed. If the NO_ECHO bit is clear, then every byte sent to the MCU must be echoed.

The start address defines where the received data is stored and where the MCU branches after the download is finished. BAF ignores the three LSB bits of download address to avoid ECC on download address. But BAF does not align the address while jumping to application for executing. So, download address should be passed as 64-bit aligned for proper execution.

The Length defines how many data bytes have to be loaded.

Table 55-25. Start address, VLE bit, NO_ECHO bit and download size in bytes

Bytes 0-3															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
START_ADDRESS[31:16]															
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
START_ADDRESS[15:0]															
Bytes 4-7															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
VLE	NO_ECHO	CODE_LENGTH[29:16]													
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CODE_LENGTH[15:0]															

55.4.8.5 Download data

Each byte of data received is downloaded into device SRAM starting from the address specified in the previous protocol step. It is not verified whether the provided address is a valid address in SRAM or is writable.

The address increments until the number of bytes of data received matches the code length specified in the previous protocol step.

Since the SRAM is protected by 64-bit wide Error Correction Code (ECC), BAF always writes to SRAM grouped in 64-bit double-words. If the last byte does not fall into a 64-bit boundary, the BAF fills it with zero’s and renders the data aligned on a 64-bit boundary. After the last data is written to SRAM, the BAF writes a dummy double-word (0000_0000_0000_0000h) to the next address location. This is written to avoid possible ECC error during core prefetch.

55.4.8.6 Execute code

The BAF program waits for the last echo message transmission being completed. Then it restores the initial MCU configuration and jumps to the loaded code at Start Address which was received in step 2 of the protocol. At this point the BAF has finished its tasks and MCU is controlled by new code executing from SRAM.

55.4.9 Serial boot configuration

55.4.9.1 LINFlexD configuration

Boot using UART protocol is implemented by the LINFlexD module.

NOTE

See the pin configuration table at the beginning of this chapter for pins used by the LINFlexD module.

When Serial Boot mode is started the LINFlexD_Rx pin is configured as an input and muxed to the LINFlexD module. The LINFlexD_Tx pin remains in a passive input state until the first byte of protocol is received. It is then configured as an output and muxed to LINFlexD module. The application may not be using the Serial Boot Mode function, and may be using the pin used by LINFlexD_Tx as an input, which may have some external hardware driving the pin. Inadvertently driving this pin as an output would cause contention and possible damage.

The LINFlexD module operates in half-duplex mode. The interface is either transmitting a byte or receiving a byte, but never both at the same time. It is expected that LINFlexD pins are connected to a physical layer bus driver/receiver (PHY). Many PHYs enforce half-duplex mode by using a single data channel for both transmitting and receiving data (such as LIN or CAN). The side effect of this is that all data transmitted on the Tx pin is also received on the Rx pin. To prevent the LINFlexD module from receiving its own transmission, the Rx pin is blocked during transmission.

The LINFlexD controller is configured to operate at a baud rate of f_{XOSC} / 832, using an 8-bit data frame with no parity bit and 1 stop bit.

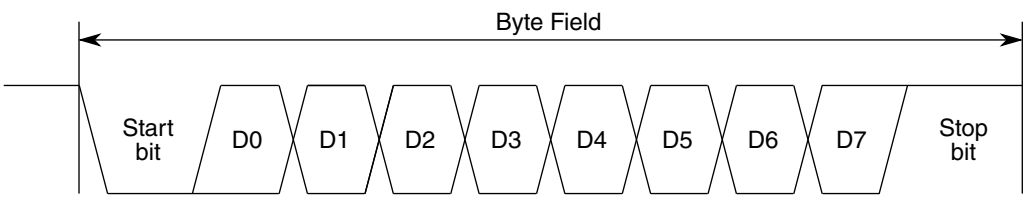


Figure 55-5. LINFlexD bit timing in UART mode

55.4.9.2 LINFlexD pin configuration

Please see the BAF configuration at the beginning of this chapter for pin details.

55.4.9.3 Baud rates

The LINFlexD baud rate is set by the frequency of an external crystal oscillator / 832. Table 55-26 shows some possible baud rates.

Table 55-26. LINFLexD baud rates

Crystal frequency	LINFlexD baud rate
8 MHz	9600 baud
12 MHz	14,400 baud
16 MHz	19,200 baud
20 MHz	24,000 baud
40 MHz	48,000 baud

55.4.9.4 Protocol summary

Table 55-27 summarizes the LINFlexD protocol and BAF action during this boot mode.

Table 55-27. LINFlexD serial boot mode download protocol

Protocol steps	Host message sent	BAF response message	Action
1	64-bit password (MSB first)	64-bit password (MSB first)	Password checked for validity and compared with stored password.
2	32-bit store address	32-bit store address	Load address is stored for future use.
3	VLE bit + NO_ECHO bit + 30 bits of download data size in bytes (MSB first)	VLE bit + NO_ECHO bit + 30 bits of download data size in bytes (MSB first)	Size of download is stored for future use. Verify VLE bit.
4	8 bits of raw binary data	If NO_ECHO bit is set, data is echoed, otherwise data is not echoed.	8 x 8 bits of data are packed into 64-bit double-words. These double-words are stored in SRAM starting from the Load address. The Load address increments until the number of data received and stored matches the size as specified in the previous steps.
5	none	none	Branch to downloaded code

55.5 Resources

- Clocks
  - Several clock configurations are made depending on the boot mode of the BAF. All settings are restored before execution of the application code (in flash memory or received via serial download).

- LINFlexD
  - The LINFlexD module is used if serial boot mode is entered. This can be through failing to find a valid boot header, or from the SER_BOOT_CBACK callback function. All settings are restored before the application code in system RAM is executed.
- PFLASH
  - PFLash module is used to disable BAF execution. This module is also used to perform Calibration RAM Remap. (see Optionally remap calibration RAM).
- Interrupts
  - No interrupts/exceptions are used or enabled, except the Machine Check exception, to handle ECC errors.
- SEMA4
  - BAF uses Semaphores when it detects multiple cores executing out of standby exit use case. Semaphores are used to synchronize and safely bring the SoC into SAFE mode.
- SRAM Controller
  - BAF sets the RAM Wait state to 1 in Standby exit when system clock is sourced from PLL.

Chapter 56
Nexus Crossbar Multi-Master Client (NXMC)

56.1 Chip-specific NXMC information

56.1.1 NXMC block connections

To provide Nexus data trace messaging from bus masters on the master ports of the crossbar (XBAR) that do not inherently have a trace client, a Nexus Crossbar Multi-Master Client (NXMC) monitors traffic into the master port of the XBAR. Trace messages transmitted over the Nexus interface (or stored in trace memory) include which NXMC generated the message as well as an identifier for the pre-concentrator source of the message.

The following figure shows a block diagram illustrating the NXMC integration.

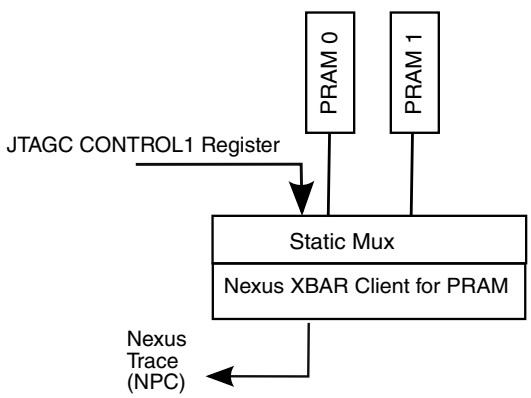


Figure 56-1. NXMC integration block diagram

56.1.2 NXMC source and master client IDs

The values of the trace master ID or in-circuit client ID, along with the Nexus message source identifier for the different clients are shown in the following table.

Table 56-1. NXMC source and master client IDs

Master Name	Master ID	Source ID
e200Z4	0000	4'b0000
e200Z2	0001	4'b0010
eDMA	0100	4'b1100
Ethernet	0101	4'b1100
FlexRAY	0110	4'b1100
Debug e200Z4	1000	4'b1100
Debug e200Z2	1001	4'b1100

56.2 Introduction

The Nexus Crossbar Multi-master Client (NXMC) module provides real-time trace capabilities in compliance with the IEEE-ISTO Nexus 5001-2012 standard. This module provides development support capabilities for devices without requiring address and data pins for internal visibility. A portion of the pin interface is also compliant with the IEEE 1149.1 JTAG standard. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port.

Many bus masters start accesses to internal address locations via the system bus. The NXMC module monitors the system bus and provides real-time trace information to debug or development tools.

56.2.1 Features

The NXMC module is compliant with the IEEE-ISTO 5001-2012 standard. The following features are implemented:

- Data trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to (selected) internal memory resources.
- Multi-master tracing capability for multiple master accesses
- HSM master(s) filtering from the tracing function based on external control
- Watchpoint messaging based on internal/external triggers, via the auxiliary pins
- Watchpoint trigger events out based on internal/external triggers
- Watchpoint trigger enable of data trace messaging
- Start or stop of Tracing (Data Trace, Watchpoint Trace) based on internal triggers.
- Optional timestamping of the messages based on timer value and controls received at the interface

- Nexus Auxiliary port for higher data throughput
  - One $\overline{\text{EVTO}}$ (Watchpoint Event) pin
  - One $\overline{\text{EVTI}}$ (Event In) pin
- Registers for data trace, watchpoint generation, and watchpoint trigger
- All features controllable and configurable via the JTAG port

56.3 External signal description

This section details information regarding the standard ports and protocol. The NXMC pin interface provides the function of transmitting messages from the messages queues to NPC. It is also responsible for handshaking with the message queues.

The NXMC module implements the following signals:

- One $\overline{\text{EVTI}}$
- One $\overline{\text{EVTO}}$

These pins are described in Table 56-3.

All NXMC input functionality is controlled through the JTAG port in compliance with IEEE 1149.1 (see IEEE 1149.1 (JTAG) Test Access Port for details). A separate JTAG TAP controller is instantiated within the NXMC. The following table describes the JTAG pins.

Table 56-2. JTAG pins for NXMC

JTAG port	Input/ output	Description
TDO	O	The Test Data Output (TDO) pin is the serial output for test instructions and data. TDO is three-stateable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK.
TDI	I	The Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK and has an internal pull-up resistor.
TMS	I	The Test Mode Select (TMS) input pin is used to sequence the JTAG test controllers' state machine. TMS is sampled on the rising edge of TCK and has an internal pull-up resistor.
TCK	I	The Test Clock (TCK) input pin is used to synchronize the test logic, and control register access through the JTAG port.
TRST	I	The Test Reset ($\overline{\text{TRST}}$) input pin is used to asynchronously initialize the JTAG controller. The $\overline{\text{TRST}}$ pin has an internal pull-down resistor.

Table 56-3. NXMC Auxiliary pins

Auxiliary port	Input/ output	Description
EVTO	O	Event Out ($\overline{\text{EVTO}}$) is an output which, when asserted, indicates one of two events has occurred based on the EOC bits in the DC Register: • one of the internal or external watchpoints has occurred and EOC = 2'b00 • system-level debug mode was entered (ipg_debug) and EOC = 2'b01 $\overline{\text{EVTO}}$ is held asserted for one cycle.
EVTI	I	Event In ($\overline{\text{EVTI}}$) is an input which initiates synchronization messages. If the Nexus module is enabled at reset, (see Enabling NXMC operation), assertion initiates data trace synchronization messages (provided data trace is enabled).

56.4 Supported messages and TCODEs

The NXMC pins allow for flexible transfer operations via public messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2012 standard defines a set of public messages. The NXMC block supports the public TCODEs listed in the following table. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 56-4. Supported public TCODEs

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 58
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 56-5)
	1	32	U-ADDR	variable	Unique portion of the data write address
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 59
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 56-5)

Table continues on the next page...

Table 56-4. Supported public TCODEs (continued)

Message name	Min packet size (bits)	Max packet size (bits)	Packet name	Packet type	Packet description
	1	32	U-ADDR	variable	Unique portion of the data read address
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	ETYPE	fixed	Error type (See Table 56-6)
	14	14	ECODE	fixed	Error code (See Table 56-7)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 60
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 56-5)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data write value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Data Trace - Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 61
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	4	4	MASTER	fixed	Indicates which master (ID) initiated the data access
	3	3	DSZ	fixed	Data size (See Table 56-5)
	1	32	F-ADDR	variable	Full access address (leading zero (0) truncated)
	8	64	DATA	fixed	Data read value (size fixed based on DSZ field)
	1	26	TSTAMP	variable	Timestamp (Optional)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	Source processor identifier (multiple Nexus configuration)
	1	8	WPHIT	variable	Watchpoint source(s) number (See Table 56-23)
	1	26	TSTAMP	variable	Timestamp (Optional)

Table 56-5. Data trace size (DSZ) encodings (TCODE = 58, 59, 60, 61)

DSZ	Transfer size
001	8-bit
010	16-bit
011	32-bit
100	64-bit
000, 101-111	Reserved

Table 56-6. Error Type (ETYPE) encodings (applicable to all error messages)

ETYPE	Description
0000	Queue overrun caused message (one or more) to be lost
0001	Contention with higher priority message caused message(s) to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus register unimplemented)
0110–1111	Reserved

Table 56-7. Error Code (ECODE) encodings (applicable based on error source)

ECODE		Description[1]
13 to 8	7 to 0	
Source (SRC)		
000000	XXXXXXX1	Watchpoint Message(s) lost
000000	XXXXXX1X	Data Trace Message(s) lost
000000	XXXXX1XX	Reserved
000000	XXXX1XXX	Reserved
000000	XXX1XXXX	Reserved
000000	XX1XXXXX	Reserved
000000	X1XXXXXX	Reserved
000000	1XXXXXXX	Reserved
Source (SRC) n = peripheral index		
CKSRCn	XXXXXXX1	Reserved
CKSRCn	XXXXXX1X	Reserved
CKSRCn	XXXXX1XX	Reserved
CKSRCn	XXXX1XXX	Reserved
CKSRCn	XXX1XXXX	Reserved
CKSRCn	XX1XXXXX	Reserved
CKSRCn	X1XXXXXX	Reserved
CKSRCn	1XXXXXXX	Reserved

1. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

56.5 Register description

This section describes the NXMC register definition. Nexus registers are accessed using the JTAG port in compliance with IEEE 1149.1. See IEEE 1149.1 (JTAG) Test Access Port for details. The complete list of registers is shown in this table.

Table 56-8. NXMC registers

Nexus register	Nexus access opcode (hex)	Read/ write	Read address (hex)	Write address (hex)
Development Control 1 (DC1)	2	R/W	04	05
Development Control 2 (DC2)	3	R/W	06	07
Watchpoint Trigger (WT)	B	R/W	16	17
Data Trace Control (DTC)	D	R/W	1A	1B
Data Trace Start Address1 (DTSA1)	E	R/W	1C	1D
Data Trace Start Address2 (DTSA2)	F	R/W	1E	1F
Data Trace End Address1 (DTEA1)	12	R/W	24	25
Data Trace End Address2 (DTEA2)	13	R/W	26	27
Breakpoint/Watchpoint Control Register1 (BWC1)	16	R/W	2C	2D
Breakpoint/Watchpoint Control Register2 (BWC2)	17	R/W	2E	2F
Breakpoint/Watchpoint Address Register1 (BWA1)	1E	R/W	3C	3D
Breakpoint/Watchpoint Address Register2 (BWA2)	1F	R/W	3E	3F

56.5.1 Development control register 1 (DC1)

The Development Control Registers control the basic development features of the NXMC module. This table shows the format of the DC1.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	EOC		0	0	WEN	0	0	0	0	0	0	0	0	0	C	0	0	0	0	0	0	0	0	0	EIC		TM		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DC1 is described in this table.

Table 56-9. DC1 register field description

Field	Description
31:29	Reserved for future functionality (Read/Write – default values must not be changed for normal operation)
28:27 EOC	$\overline{\text{EVTO}}$ Control. 00 $\overline{\text{EVTO}}$ upon occurrence of watchpoint (internal) 01 $\overline{\text{EVTO}}$ upon entry into system-level Debug Mode 1x Reserved
26:25	Reserved for future functionality (read as 0)
24 WEN	Watchpoint Trace Enable. 0 Watchpoint Messaging disabled 1 Watchpoint Messaging enabled
23:5	Reserved for future functionality (read as 0)
4:3 EIC	$\overline{\text{EVTI}}$ Control. 00 $\overline{\text{EVTI}}$ for synchronization (Data Trace) 01 Reserved 10 $\overline{\text{EVTI}}$ disabled 11 Reserved
2:0 TM	Trace Mode. 000 No Trace 1xx Reserved x1x Data Trace enabled xx1 Reserved

56.5.2 Development control register 2 (DC2)

The Development Control Registers control the basic development features of the NXMC module. This table shows the format of the DC2.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EWC								0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DC2 is described in this table.

Table 56-10. DC2 register field description

Field	Description	
31:24	$\overline{\text{EVTO}}$ Watchpoint Configuration[1].	
EWC	00000000	No Watchpoints trigger $\overline{\text{EVTO}}$
	1XXXXXXX	Reserved (This is read/write field, so default value must not be changed for normal operation)
	X1XXXXXX	Reserved (This is read/write field, so default value must not be changed for normal operation)
	XX1XXXXX	Reserved (This is read/write field, so default value must not be changed for normal operation)
	XXX1XXXX	Reserved (This is read/write field, so default value must not be changed for normal operation)
	XXXX1XXX	Internal Watchpoint #1 triggers $\overline{\text{EVTO}}$
	XXXXX1XX	Internal Watchpoint #2 triggers $\overline{\text{EVTO}}$
	XXXXXX1X	Reserved (This is read/write field, so default value must not be changed for normal operation)
	XXXXXXX1	Reserved (This is read/write field, so default value must not be changed for normal operation)
23:1	Reserved for future functionality (read as 0)	
0	Reserved for future functionality (Read/Write – default values must not be changed for normal operation)	

1. The EOC bits in DC1 must be programmed to trigger EVTO on watchpoint occurrence for the EWC bits to have any effect.

56.5.3 Watchpoint trigger register (WT)

The Watchpoint Trigger Register allows the watchpoints defined either internally to the NXMC module or by an external module to trigger actions. These watchpoints can control data trace enable and disable.

The WT bits can be used to produce an address related window for triggering trace messages. This table shows the format of the WT register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	DTS			DTE			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The WT register is described in this table.

Table 56-11. WT register field description

Field	Description[1]
31:26	Reserved for future functionality (read as 0)
25:23 DTS	Data Trace Start Control. 000 Trigger disabled 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved
22:20 DTE	Data Trace End Control. 000 Trigger disabled 001 Reserved 010 Reserved 011 Reserved 100 Reserved 101 Use Internal Watchpoint #1 (BWA1 Register) 110 Use Internal Watchpoint #2 (BWA2 Register) 111 Reserved
19:0	Reserved for future functionality (read as 0)

1. The WT bits only enable data trace if the TM bits in the Development Control Register (DC) have not already been set to enable data trace.

56.5.4 Data trace control register (DTC)

The Data Trace Control Register controls whether DTM messages are restricted to reads, writes or both for a user programmable address range. There are two data trace channels controlled by the DTC for the NXMC module. This table shows the format of the DTC register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RWT1		RWT2		0	0	0	ME	0	0			MID				0	0	0	0	0	0	0	0	RC1	RC2	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The DTC register is described in this table.

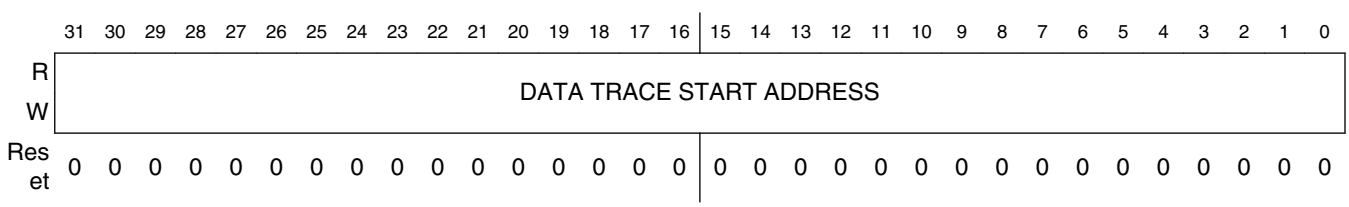
Table 56-12. DTC field description

Field	Description
31:30 RWT1	Read/Write Trace 1. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace
29:28 RWT2	Read/Write Trace 2. 00 No trace messages generated x1 Enable data read trace 1x Enable data write trace
27:25	Reserved for future functionality
24 MME	Multi Master Tracing enable. 0 Tracing enabled only for Master with ID = MID (DTC[19:16]) 1 Tracing enabled for all the Masters
23:20	Reserved for future functionality
19:16 MID	ID of the system bus Master to be traced[1]
15:8	Reserved for future functionality
7 RC1	Range Control 1. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
6 RC2	Range Control 2. 0 Condition trace on address within range (endpoints inclusive) 1 Condition trace on address outside of range (endpoints exclusive)
5:0	Reserved for future functionality

1. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

56.5.5 Data trace start address registers 1 and 2 (DTSA1 and DTSA2)

The Data Trace Start Address Registers define the start addresses for each trace channel. This table shows the format of the DTSA1 and DTSA2 registers.



56.5.6 Data trace end address registers 1 and 2 (DTEA1 and DTEA2)

The Data Trace End Address Registers define the end addresses for each Trace channel. This table shows the format of the DTEA1 and DTEA2 registers.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA TRACE END ADDRESS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This table illustrates the range that are selected for data trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 56-13. Data trace address range options

Programmed values	Range control bit value	Range selected
DTSA ≤ DTEA	0	DTSA -> <- DTEA
DTSA ≤ DTEA	1	<- DTSA DTEA ->
DTSA > DTEA	N/A	Invalid range—no trace

Note

DTSA must be less than (or equal to) DTEA in order to guarantee correct data write/read traces.

When the range control bit is 0 (internal range), accesses to DTSA and DTEA addresses are traced. When the range control bit is 1 (external range), accesses to DTSA and DTEA are not traced.

56.5.7 Breakpoint/watchpoint control register 1 (BWC1)

Breakpoint/watchpoint control register 1 controls the attributes for generation of NXMC watchpoint#1. This table shows the format of the BWC1 register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BWE1		BRW1		0	0	0	0	0	0	0	0	0	0	BWR1		BWT1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The BWC1 register is described in this table.

Table 56-14. BWC1 field description

Field	Description	
31:30	Breakpoint/Watchpoint #1 Enable.	
BWE1	00	Internal Nexus Watchpoint #1 disabled
	01	Reserved
	10	Reserved
	11	Internal Nexus Watchpoint #1 enabled
29:28	Breakpoint/Watchpoint #1 Read/Write Select.	
BRW1	00	Watchpoint #1 hit on read accesses
	01	Watchpoint #1 hit on write accesses
	10	Watchpoint #1 on read or write accesses
	11	Reserved
27:18	Reserved for future functionality (read as 0)	
17:16	Breakpoint/Watchpoint #1 Register Compare.	
BWR1	00	No Register Compare (same as BWC1[31:30] = 2'b00)
	01	Reserved
	10	Compare with BWA1 value
	11	Reserved
15	Breakpoint/Watchpoint #1 Type.	
BWT1	0	Reserved
	1	Watchpoint #1 on data accesses
14:0	Reserved for future functionality (read as 0)	

56.5.8 Breakpoint/watchpoint control register 2 (BWC2)

Breakpoint/Watchpoint Control Register2 controls attributes for generation of NXMC watchpoint#2. This table shows the format of the BWC2 register.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BWE2		BRW2		0	0	0	0	0	0	0	0	0	0	BWR2		BWT2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The BWC2 register is described in this table.

Table 56-15. BWC2 field description

Field	Description
31:30 BWE2	Breakpoint/Watchpoint #2 Enable. 00 Internal Nexus Watchpoint #2 disabled 01 Reserved 10 Reserved 11 Internal Nexus Watchpoint #2 enabled
29:28 BRW2	Breakpoint/Watchpoint #2 Read/Write Select. 00 Watchpoint #2 hit on read accesses 01 Watchpoint #2 hit on write accesses 10 Watchpoint #2 on read or write accesses 11 Reserved
27:18	Reserved for future functionality (read as 0)
17:16 BWR2	Breakpoint/Watchpoint #2 Register Compare. 00 No Register Compare (same as BWC2[31:30] = 2b00) 01 Reserved 10 Compare with BWA2 value 11 Reserved
15 BWT2	Breakpoint/Watchpoint #2 Type. 0 Reserved 1 Watchpoint #2 on data accesses
14:0	Reserved for future functionality (read as 0)

56.5.9 Breakpoint/watchpoint address registers 1 and 2 (BWA1 and BWA2)

The Breakpoint/Watchpoint Address Registers are compared with system bus addresses in order to generate internal watchpoints. This table shows the format of the BWA1 and BWA2 registers.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BREAKPOINT / WATCHPOINT ADDRESS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.10 Unimplemented registers

Unimplemented registers are those with client select and index value combinations other than those listed in Table 56-8. For unimplemented registers, the NXMC module drives TDO to zero during the SHIFT-DR state. It also transmits an error message with the invalid access opcode encoding (ETYPE = 0101).

56.6 Programming considerations (RESET)

If NXMC register configuration is to occur during system reset (as opposed to debug mode), all NXMC configuration should be completed between the negation of $\overline{\text{TRST}}$ and system reset de-assertion. The JTAG ID register should be read by the tool after negation of $\overline{\text{TRST}}$ and should be programmed before deassertion of system reset.

56.6.1 IEEE 1149.1 (JTAG) Test Access Port

The NXMC module uses the IEEE 1149.1 TAP controller for accessing Nexus resources. The JTAG signals themselves are shared by all TAP controllers on the device. The NXMC module implements a 4-bit Instruction Register (IR). The valid instructions and method for register access are outlined in NXMC register access via JTAG.

56.6.2 Enabling NXMC operation

The Nexus module is enabled by loading a single instruction into the JTAG Instruction Register (IR). Once enabled, the module is ready to accept control input via the JTAG pins.

The Nexus module is disabled when the JTAG state machine reaches the Test-Logic-Reset state. This state can be reached by the assertion of the $\overline{\text{TRST}}$ pin or by cycling through the state machine using the TMS pin. The Nexus module is also disabled if a Power-on-Reset (POR) event occurs.

If the NXMC module is disabled, no trace output is provided. Nexus registers are not available for reads or writes.

56.6.3 Enabling NXMC TAP controller

Assertion of a Power-on-Reset signal or assertion of the $\overline{\text{TRST}}$ pin resets all TAP controllers. Upon exit from the Test-Logic-Reset state, the IR value is loaded with the JTAG ID. When the NXMC TAP is accessed, this information helps the development tool obtain information about the Nexus module it is accessing, such as version, sequence, feature set, etc.

56.6.4 NXMC register access via JTAG

Access to Nexus register resources is enabled by loading a single instruction into the JTAG Instruction Register (IR).

Once the JTAG NEXUS-ACCESS instruction has been loaded, the JTAG port allows tool/target communications with all Nexus registers according to the map in Table 56-8.

Reading/writing of a Nexus register then requires two passes through the Data-Scan (DR) path of the JTAG state machine.

1. The first pass through the DR selects the Nexus register to be accessed by providing an index (see Table 56-8), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the format shown below.

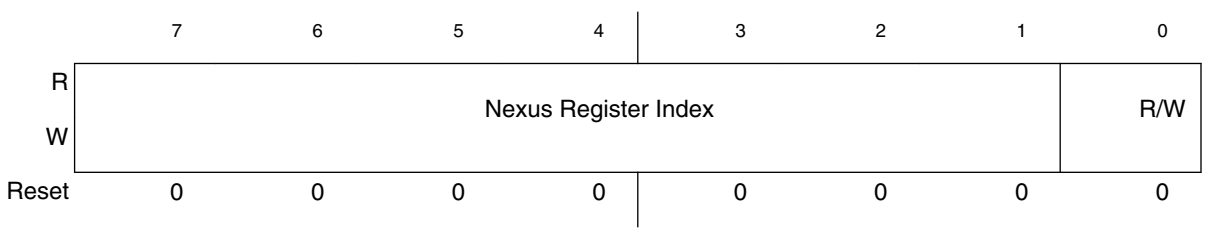


Table 56-16. JTAG DR field description for Nexus register access

Field	Description
Nexus Register Index	Selected from values in Table 56-8
Read/Write (R/W)	0 Read 1 Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.

a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the Capture-DR state.

b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the Update-DR state.

56.7 Functional description

The following sections describe the functionality of the NXMC module.

56.7.1 Data trace

This section describes the data trace mechanism supported by the NXMC module. Data trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the IEEE-ISTO 5001-2012 standard.

56.7.1.1 Data trace messaging (DTM)

NXMC data trace messaging is accomplished by snooping the system bus and storing the information for qualifying accesses (based on enabled features and matching target addresses). The NXMC module traces all data access that meet the selected range and attributes.

56.7.1.2 DTM message formats

The NXMC block supports five types of DTM messages:

- Data write
- Data read
- Data write synchronization
- Data read synchronization
- Error messages

56.7.1.2.1 Data write and data read messages

The data write and data read messages contain the data write/read value and the address of the write/read access, relative to the previous data trace message. The DATA field is considered fixed based on the DSZ encodings to either 8, 16, 32 , 64. The TSTAMP field is 26 bits. Data write message and data read message information is messaged out in the format shown in the following figure. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

Table 56-17. DTM message format for DATA width = 64 bits

8/16/32/64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	U-ADDR	DSZ	MASTER	SRC	TCODE (111010 or 111011)

TSTAMP is another optional field which is inserted based on the controls received from the NPC, see Timestamping feature.

56.7.1.2.2 Data trace with non-contiguous byte strobes asserted

The v6 extensions to AMBA 2.0 allow system bus transfers with bytes non-asserted byte strobes in between asserted byte strobes. The NXMC does not support transfers with non-contiguous byte strobes asserted. In such a condition, the invalid bytes in the DATA packet are driven to the value of 0xFF.

56.7.1.2.3 DTM overflow error messages

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards incoming messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a data trace message attempts to enter the queue while it is being emptied, the error message incorporates the data trace only error encoding (ECODE = 00000000000010). If a watchpoint also attempts to be queued while the FIFO is being emptied, then the error message incorporates error encoding (ECODE = 00000000000011).

Error information is messaged out in the format shown in this figure. The SRC field values are described in the device-specific chapter that describes how the modules are configured.

Table 56-18. DTM error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

NOTE

The internal queue FIFO is very small. This causes FIFO overflows whenever Trace is enabled on any of the AHB clients and the AHB master module generates >4 burst accesses. If the customer is experiencing a lot of overflow error messages the amount of data snooped by the NXMC should be reduced.

56.7.1.2.4 Data trace synchronization messages

A data trace write/read with synchronization message is messaged via the auxiliary port (provided data trace is enabled) for the following conditions (see Table 56-20):

- Initial data trace message upon exit from system reset or whenever data trace is enabled is a synchronization message.
- Upon returning from a low power state, the first data trace message is a synchronization message.
- Upon returning from debug mode, the first data trace message is a synchronization message.
- After occurrence of queue overrun (can be caused by any trace message), the first data trace message is a synchronization message.
- After the periodic data trace counter has expired indicating 255 without-sync data trace messages have occurred since the last with-sync message occurred.
- Upon assertion of the event in (EVTI) pin, the first data trace message is a synchronization message if the EIC bits of the DC Register have enabled this feature.
- Upon data trace write/read after the previous DTM Message was lost due to a collision while entering the FIFO between the DTM message and any of the following: error message, or watchpoint message or peripheral message. This is a very rare occurrence and possible only in case of continuous burst on the system bus as well as watchpoint events which is unrealistic.

Data trace synchronization messages provide the full address (without leading zeros) and ensure that development tools fully synchronize with data trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the data trace address is transmitted. The format for data trace write/read with synchronization messages is shown in the following figure. The SRC and MASTER field values are described in the device-specific chapter that describes how the modules are configured.

Table 56-19. Data write/read with synchronization message format

8-64 bits	1–32 bits	3 bits	4 bits	4 bits	6 bits
DATA	F-ADDR	DSZ	MASTER	SRC	TCODE (111100 or 111101)

Exception conditions that result in data trace synchronization are summarized in this table.

Table 56-20. Data trace exception summary

Exception condition	Exception handling
System Reset Negation	At the negation of JTAG reset ($\overline{\text{TRST}}$), queue pointers, counters, state machines, and registers within the NXMC module are reset. If data trace is enabled, the first data trace message is a data write/read with synchronization message.
Data Trace Enabled	The first data trace message (after data trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a low power mode or debug mode the next data trace message is converted to a data write/read with synchronization message.
Queue Overrun	An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue is a data write/read with synchronization message.
Periodic Data Trace Synchronization	A forced synchronization occurs periodically after 255 data trace messages have been queued. A data write/read with synchronization message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, an $\overline{\text{EVTI}}$ assertion initiates a data write/read with synchronization message upon the next data write/read (if data trace is enabled and the EIC bits of the DC Register have enabled this feature).
Collision Priority	All messages have the following priority: Error > WPM > DTM. Although a very rare occurrence, during simultaneous burst lower priority messages can be lost due to burst of high priority messages. Proper error messages are generated and in case of DTM lost a subsequent read/write queues a data write/read with synchronization message.

56.7.1.3 Enabling data trace messaging operation

Data trace messaging can be enabled in one of three ways:

- Setting the TM field of the DC Register via JTAG to enable data trace (DC[1]).

- Using the DTS field of the WT Register to enable data trace on watchpoint hits (either watchpoints configured within the NXMC module or external watchpoint inputs controlled by the device).
- By toggling the external start control, which in turn sets the TM field of the DC register to enable data trace. (The TM bit clears and hence the data tracing stops on the toggle of external stop control. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note

When enabling the DTM, either via JTAG or external start/stop control, it is important to make sure that all the other controls (i.e. DTC, DTSA/E1/2 and WT register fields) are correctly configured to ensure correct DTM functionality.

56.7.1.4 DTM queueing

The NXMC implements a programmable depth queue for queuing all messages. Messages that enter the queue are transmitted via the auxiliary pins in the order in which they are queued.

Note

If multiple trace messages need to be queued at the same time, watchpoint messages have the highest priority (WPM > DTM).

56.7.1.5 Relative addressing

The relative address feature is compliant with IEEE-ISTO 5001-2012 and is designed to reduce the number of bits transmitted for addresses of data trace messages. The address transmitted is relative to the address of the previous data trace message. It is generated by XORing the new address with the previous address, and then using only the results up to the most significant 1 in the result. To recreate this address, an XOR of the (most-significant 0-padded) message address with the previously decoded address gives the current address.

For example, if the previous address (A1) = 0x0003FC01, the new address (A2) = 0x0003F365 as shown in this figure.

Message Generation:

A1 = 0000 0000 0000 0011 1111 1100 0000 0001
A2 = 0000 0000 0000 0011 1111 0011 0110 0101

A1 ⊕ A2 = 0000 0000 0000 0000 0000 1111 0110 0100

Address Message (M1) = 1111 0110 0100

Address Re-creation:

A1 ⊕ M1 = A2
A1 = 0000 0000 0000 0011 1111 1100 0000 0001
M1 = 0000 0000 0000 0000 0000 1111 0110 0100

A2 = 0000 0000 0000 0011 1111 0011 0110 0101

Figure 56-2. Relative addressing example

56.7.1.6 Data trace windowing

Data write/read messages are enabled via the RWT1(2) field in the Data Trace Control Register (DTC) for each DTM channel. Data trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC1(2) field in the DTC. All system bus DMA initiated read/write accesses which fall inside or outside these address ranges, as programmed, are candidates to be traced.

56.7.1.7 System bus cycle special cases

The system bus cycle special cases are shown in this table.

Table 56-21. System Bus cycle special cases

Special case	Action
System bus cycle with data error	Data trace message discarded
System bus cycle completed without error	Cycle captured and transmitted
System bus cycle is an instruction fetch	Cycle ignored

56.7.1.8 System bus multi-master tracing

DTM tracing can be done either for all the masters or for a single master (indicated by the MID field of the DTC register) based on the DTC register MME field. In case of single master, the correct value needs to be configured in the MID register before enabling the data trace operation.

The MASTER field of the DTM message (see figure in Data write and data read messages for an example) indicates the system bus master for which the trace has been generated. For each masters (0 to 15) there is a parameter associated (MSID0-15) for indicating its actual ID, which finally gets inserted in the MASTER field. The MASTER field values are described in the device-specific chapter that describes how the modules are configured.

56.7.1.9 HSM master filtering

For security reasons, certain HSM masters need to be disabled or filtered from data tracing operations.

56.7.2 Watchpoint support

The NXMC module provides watchpoint messaging via the auxiliary pins, as defined by IEEE-ISTO 5001-2012. Watchpoint messages can be generated by either using the NXMC defined internal watchpoints or by externally defined watchpoint signals.

56.7.2.1 Watchpoint messaging

Enabling watchpoint messaging is accomplished by setting the WEN bit in the DC Register. This bit is set either via JTAG write 1 access or external start indication, and cleared via JTAG write 0 access or external start indication. In case of conflict between JTAG write and external start/stop control, JTAG write takes precedence.

Note

In case of enabling the watchpoint messaging, either via JTAG or external start/stop control, it is important to make sure that all the other Watch Point Message (WPM) controls (i.e. BWC1/2, BWA1/2 register fields) are configured with the correct values in order to ensure correct WPM operation.

Watchpoint setting is supported through two methods:

- Internal watchpoints—Using the BWC1(2) registers, two independently controlled internal watchpoints can be initialized. When a system bus access address matches on BWA1(2), a watchpoint message is transmitted. The system bus multi-master or single master controls applies here too as explained for DTM (in System bus multi-master tracing).
- External watchpoints—The NXMC module supports two external watchpoint inputs. The optional logic to generate these watchpoint signals is implemented at the device level. The watchpoints are described in the device-specific chapter that describes how the modules are configured. If either of these signals asserts (i.e, on 0-to-1 transition), a watchpoint message is transmitted.

The Nexus module provides watchpoint messaging using the IEEE-ISTO 5001 defined TCODE. When any of the four possible watchpoint sources asserts, a message is sent to the queue to be messaged out. This message indicates the watchpoint number as shown in Table 56-22 and Table 56-23.

Table 56-22. Watchpoint message format (fixed length = 18 bits)

8 bits	4 bits	6 bits
WPHIT (RRRRXXXX)	SRC	TCODE (001111)

Table 56-23. Watchpoint source description

Watchpoint source (8-bits)	Watchpoint description
RRRRXXX1[1]	External Watchpoint #1(device defined)
RRRRXX1X	External Watchpoint #2 (device defined)
RRRRX1XX	Internal Watchpoint #1 (BWA1 match)
RRRR1XXX	Internal Watchpoint #2 (BWA2 match)

1. R = reserved for future use (insert 0s).

56.7.2.2 Watchpoint error message

An error message occurs when a new message cannot be queued due to the message queue being full. The FIFO discards messages until it has completely emptied the queue. Once emptied, an error message is queued. The error encoding indicates which type(s) of messages attempted to be queued while the FIFO was being emptied.

If only a watchpoint message attempts to enter the queue while it is being emptied, the error message incorporates the watchpoint only error encoding (ECODE = 00000000000001). If a data trace message also attempts to enter the queue while it is being emptied, the error message incorporates error encoding (00000000000011). Error information is messaged out in the format shown in this figure.

Table 56-24. Error message format (fixed length = 28 bits)

14 bits	4 bits	4 bits	6 bits
ECODE	ETYPE	SRC	TCODE (001000)

56.7.3 Timestamping feature

This feature allows a timestamp packet to be appended to all messages, based on the timer value and controls received externally from the NPC. If enabled, this packet is added as the last field to all the outgoing messages. The TSTAMP packet width is 26 bits.

The TSTAMP packet is appended at the end of a message during message formatting, before transfer. This figure shows an example of ICTM message with a TSTAMP field.

Table 56-25. Message with TSTAMP field example

26 bits	1–44 bits	6 bits	4 bits	6 bits
TSTAMP	CKDATAn	CKSRCn	SRC	TCODE (100010)

The enable or disable indication should remain static for the complete debug session. Dynamically enabling or disabling of the timestamping is not allowed during running debug tracing. The following steps allow seamless enabling/disabling of the timestamping:

1. All tracing (DTM/WPM/ICTM) must be disabled
2. Wait for the internal queues to be emptied (no further request is asserted at the auxiliary interface)
3. Change the timestamp enable indication to assert/deassert
4. Enable the required tracing controls (via TM, PME0 etc. control bits)

Chapter 57
JTAG Controller (JTAGC)

57.1 Chip-specific JTAG information

57.1.1 JTAG Controller (JTAGC) configuration

The device contains one JTAGC module. The JTAGC is the Nexus1 interface between the debugger tool and other Nexus1 clients on the device. The JTAGC implements a number of instructions to enable debugger access to those other clients, as described in the following sections.

The JTAGC CONTROL1 register is used to control timestamping and select which PRAM AHB transactions are monitored, as described below.

57.1.2 JTAGC Control 1 register

The JTAGC Control 1 register is a 32-bit shift register path from TDI to TDO selected when the ENABLE_CONTROL1 instruction is active. The default reset value of the JTAGC Control 1 register is 0. The Control 1 register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	PAD CTRL	RESERVED		TENX MC	TMZ2		TEZ2	0			TMZ4		TEZ4	SNOOP	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The following table describes the JTAGC Control 1 register functions.

Table 57-1. JTAGC Control 1 register field descriptions

Field	Description
PADCTRL	Disable the NPC port enables so that SIUL can control the pads. 0 NPC port enables control the npc pads 1 Control transferred to SIULd
TENXMC	NXMC Timestamp Enable. 0 Timestamping is disabled 1 Timestamping is enabled
TMZ2	Timestamp Mode for Z2 core. Defines Timestamping granularity. 00 Add timestamp to every message 01 Add timestamp to every 4th message 10 Add timestamp to every 16th message 11 Add timestamp to every 64th message
TEZ2	Z2 core Timestamp Enable. 0 Timestamping is disabled 1 Timestamping is enabled
TMZ4	Timestamp Mode for Z4 core. Defines Timestamping granularity. 00 Add timestamp to every message 01 Add timestamp to every 4th message 10 Add timestamp to every 16th message 11 Add timestamp to every 64th message
TEZ4	Z4 core Timestamp Enable. 0 Timestamping is disabled 1 Timestamping is enabled
SNOOP	Used to select the pram AHB bus for snooping. 00 No snooping (Default) 01 Snooping of PRAM0 AHB transactions 10 Snooping of PRAM1 AHB transactions 11 No snooping (Default)

57.2 Introduction

The JTAGC block provides the means to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from the JTAGC block is communicated in serial format.

57.2.1 Block diagram

The following is a simplified block diagram of the JTAG Controller (JTAGC) block. Refer to the chip-specific configuration information as well as Register description for more information about the JTAGC registers.

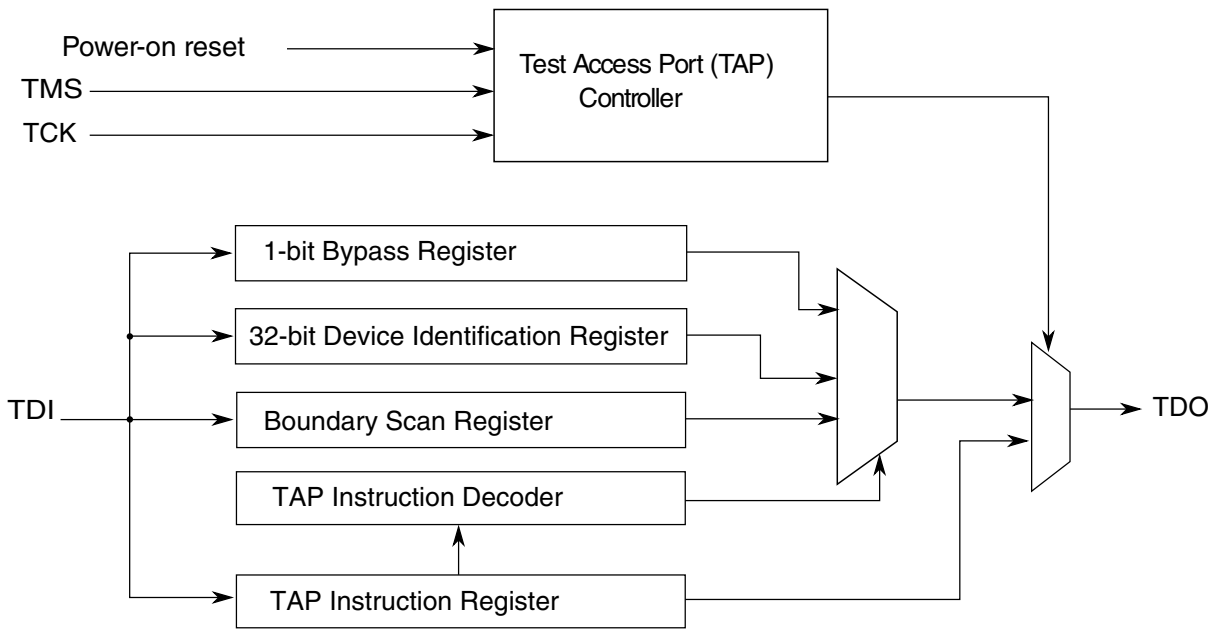


Figure 57-1. JTAG (IEEE 1149.1) block diagram

57.2.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 Test Access Port (TAP) interface
  - 4 pins (TDI, TMS, TCK, and TDO)
- Instruction register that supports several IEEE 1149.1-2001 defined instructions as well as several public and private device-specific instructions. Refer to Table 57-5 for a list of supported instructions.
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- JTAG_PASSWORD register
- Control 1 register

- Bypass register, boundary scan register, data registers, and device identification register.
- TAP controller state machine that controls the operation of the data registers, instruction register and associated circuitry.

57.2.3 Modes of operation

The JTAGC block uses a power-on reset indication as its primary reset signals. Several IEEE 1149.1-2001 defined test modes are supported, as well as a bypass mode.

57.2.3.1 Reset

The JTAGC block is placed in reset when either power-on reset is asserted, or the TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state. Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset results in asynchronous entry into the reset state. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the on-chip system logic to continue unhindered
- The instruction register is loaded with the IDCODE instruction

57.2.3.2 IEEE 1149.1-2001 defined test modes

The JTAGC block supports several IEEE 1149.1-2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register while the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, SAMPLE and SAMPLE/PRELOAD. Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic while the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the BYPASS, HIGHZ, CLAMP or reserved instructions are active. The functionality of each test mode is explained in more detail in JTAGC block instructions.

57.2.3.3 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. While in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

57.3 External signal description

The JTAGC consists of a set of signals that connect to off chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 57-2. JTAG signal properties

Name	I/O	Function	Reset State
TCK	Input	Test Clock	Weak pullup
TDI	Input	Test Data In	Weak pullup
TDO	Output	Test Data Out	High Z[1]
TMS	Input	Test Mode Select	Weak pullup

1. TDO output buffer enable is negated when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

57.3.1 TCK—Test clock input

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

57.3.2 TDI—Test data input

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

57.3.3 TDO—Test data output

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is three-stateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in TAP controller state machine.

57.3.4 TMS—Test mode select

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

57.4 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

57.4.1 Instruction register

The JTAGC block uses a 6-bit instruction register as shown in the following figure. The instruction register allows instructions to be loaded into the block to select the test to be performed or the test data register to be accessed or both. Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 000001b, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	5	4	3	2	1	0
R	0	0	0	0	0	1
W	Instruction Code					
Reset:	0	0	0	0	0	1

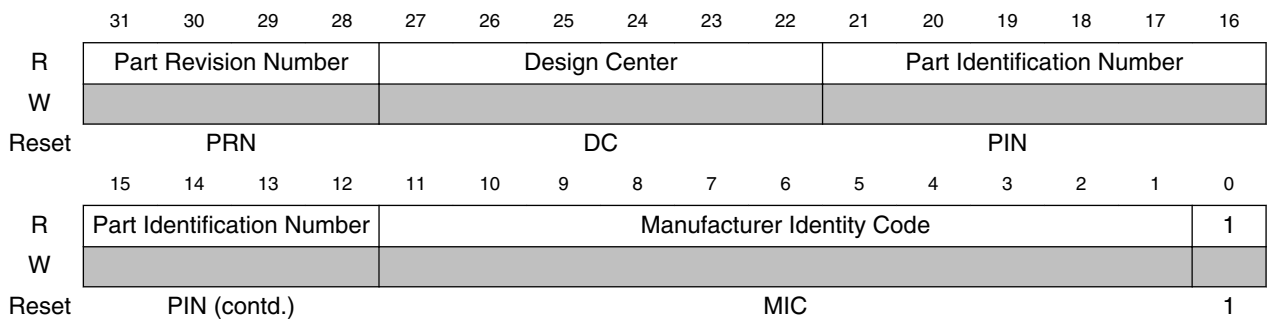
Figure 57-2. Instruction register

57.4.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS, CLAMP, HIGHZ or reserve instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

57.4.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP. The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state while the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.



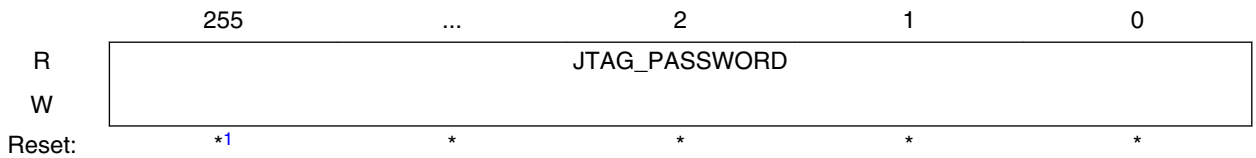
The following table describes the device identification register functions.

Table 57-3. Device identification register field descriptions

Field	Description
PRN	Part Revision Number. Contains the revision number of the part. Value is 0x1.
DC	Design Center. Indicates the design center. Value is 0x26.
PIN	Part Identification Number. Contains the part number of the device. 0x84.
MIC	Manufacturer Identity Code. Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID. Value is 0x00E.
IDCODE ID	IDCODE Register ID. Identifies this register as the device identification register and not the bypass register. Always set to 1.

57.4.4 JTAG_PASSWORD register

The JTAG_PASSWORD register is a 256-bit shift register path from TDI to TDO selected when the ENABLE_JTAG_PASSWORD instruction is active. The default reset value of the JTAG_PASSWORD register is 256'b0. The JTAG_PASSWORD register transfers its value to a parallel hold register on the rising edge of TCK when the TAP controller state machine is in the Update-DR state. Once the ENABLE_JTAG_PASSWORD instruction is executed, the register value remains valid until a JTAG reset occurs. The operation of this register is described in the security documentation.



1. The reset value of JTAG_PASSWORD is 256 'b0.

The following table describes the JTAG_PASSWORD register functions.

Table 57-4. JTAG_PASSWORD register field descriptions

Field	Description
JTAG_PASSWORD	JTAG Password. The JTAG_PASSWORD bits are used to provide the JTAG password for security.

57.4.5 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, SAMPLE or SAMPLE/PRELOAD instructions are active. It is used to capture input pin data, force fixed values on output pins, and select a logic value and direction for bidirectional pins. Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in Boundary scan. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

57.5 Functional description

This section explains the JTAGC functional description.

57.5.1 JTAGC reset configuration

While in reset, the TAP controller is forced into the Test-Logic-Reset state, thus disabling the test logic and allowing normal operation of the on-chip system logic. In addition, the instruction register is loaded with the IDCODE instruction.

57.5.2 IEEE 1149.1-2001 (JTAG) Test Access Port

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions refer to ACCESS_AUX_x instructions.

Data is shifted between TDI and TDO though the selected register starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register, test data registers, and the bypass register.

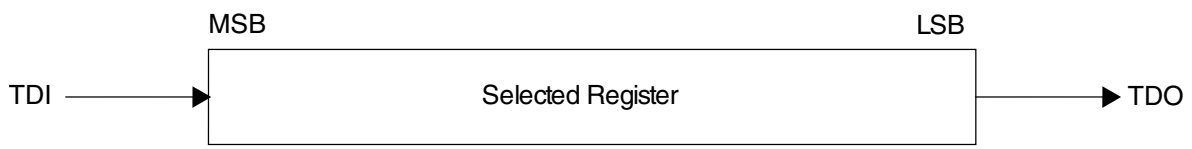
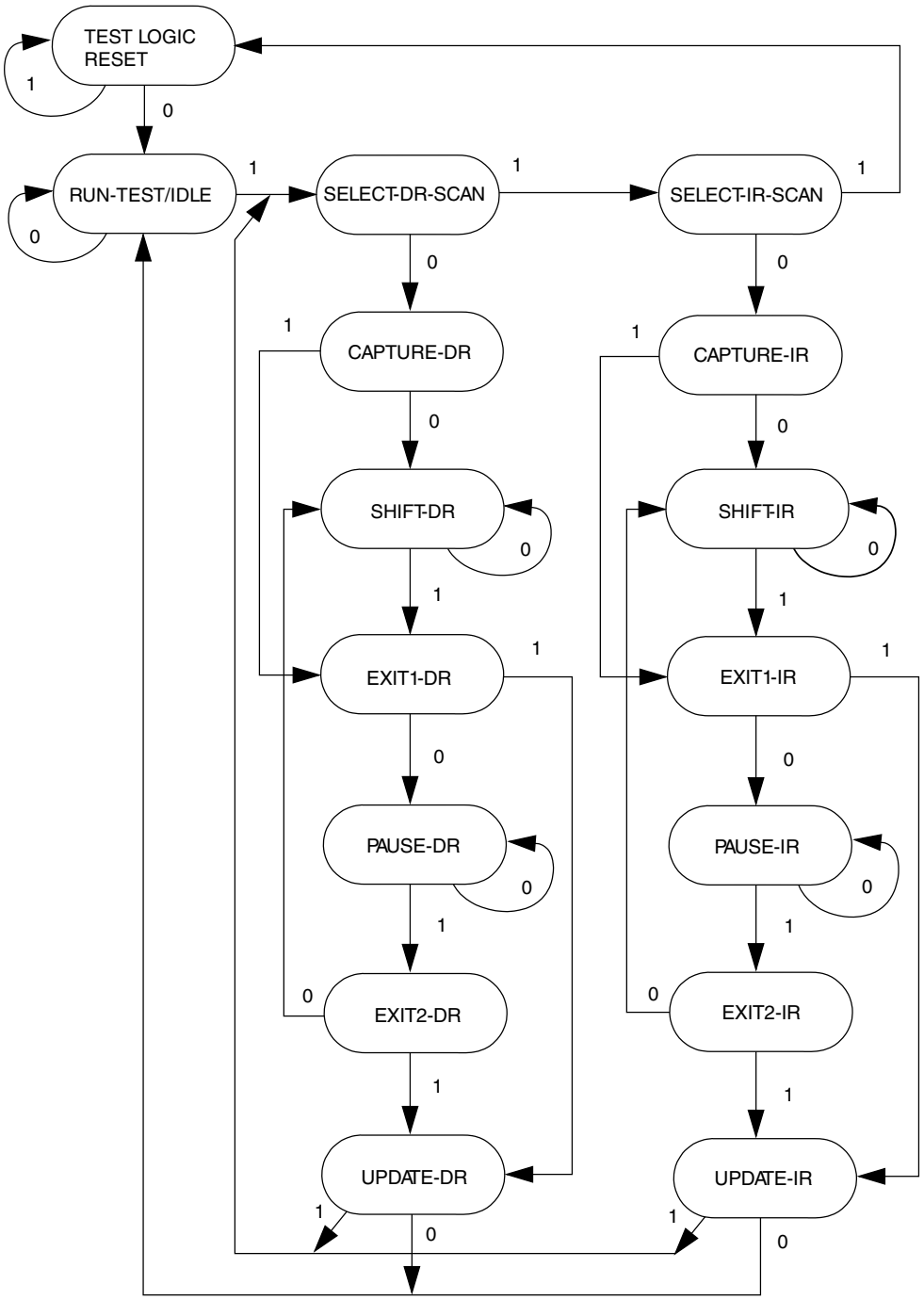


Figure 57-3. Shifting data through a register

57.5.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin. The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the following figure shows, holding TMS at logic 1 while clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.



The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 57-4. IEEE 1149.1-2001 TAP controller finite state machine

57.5.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting the JTAGC enable to a logic 1 value.

57.5.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions while the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

57.5.4 JTAGC block instructions

The JTAGC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 57-5. General 6-bit JTAG instructions

Instruction	Code[5:0]	Instruction Summary
IDCODE	000001	Selects device identification register for shift
SAMPLE/PRELOAD	000010	Selects boundary scan register for shifting, sampling, and preloading without disturbing functional operation
SAMPLE	000011	Selects boundary scan register for shifting and sampling without disturbing functional operation
EXTEST	000100	Selects boundary scan register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.
Factory debug reserved	000101	Intended for factory debug only
Factory debug reserved	000110	Intended for factory debug only
ENABLE_JTAG_PASSWORD	000111	Selects JTAG_PASSWORD register
HIGHZ	001001	Selects bypass register and three-states all output pins. **NOTE:** Execution of this instruction asserts functional reset.
Factory debug reserved	001010	Intended for factory debug only
CLAMP	001100	Selects bypass register and applies preloaded values to output pins. **NOTE:** Execution of this instruction asserts functional reset.

Table continues on the next page...

Table 57-5. General 6-bit JTAG instructions (continued)

Instruction	Code[5:0]	Instruction Summary
ENABLE_CONTROL1	001110	Enables access to Control 1 register
Reserved	001111	Reserved
BYPASS	111111	Selects bypass register for data operations
Reserved[1]	All other opcodes	Decoded to select bypass register
ACCESS_AUX_x	100000-111110	Grants one of the auxiliary TAP controllers ownership of the TAP

1. The manufacturer reserves the right to change the decoding of reserved instruction codes in the future

The ACCESS_AUX instructions are described in the chip configuration debug information.

57.5.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

57.5.4.2 SAMPLE/PRELOAD instruction

The SAMPLE/PRELOAD instruction has two functions:

- The SAMPLE portion of the instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE/PRELOAD instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. Both the data capture and the shift operation are transparent to system operation.
- The PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

57.5.4.3 SAMPLE instruction

The SAMPLE instruction obtains a sample of the system data and control signals present at the MCU input pins and just before the boundary scan register cells at the output pins. This sampling occurs on the rising edge of TCK in the Capture-DR state when the SAMPLE instruction is active. The sampled data is viewed by shifting it through the boundary scan register to the TDO output during the Shift-DR state. There is no defined action in the Update-DR state. Both the data capture and the shift operation are transparent to system operation.

57.5.4.4 EXTEST External test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the preloaded data is loaded into the boundary scan register using the SAMPLE/PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state while performing external boundary scan operations.

57.5.4.5 ENABLE_JTAG_PASSWORD instruction

The ENABLE_JTAG_PASSWORD instruction selects the JTAG_PASSWORD register for connection as the shift path between TDI and TDO.

57.5.4.6 ENABLE_CONTROL1 instruction

The ENABLE_CONTROL1 instruction selects the Control 1 register for connection as the shift path between TDI and TDO.

57.5.4.7 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. While HIGHZ is active all output drivers are placed in an inactive drive state (e.g., high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

57.5.4.8 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register while the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) while conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

57.5.4.9 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers. When this instruction is loaded, control of the JTAG pins are transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins. The JTAGC regains control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE while they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

57.5.4.10 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. While the BYPASS instruction is active the system logic operates normally.

57.5.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, SAMPLE, or SAMPLE/PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

57.6 Initialization/Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Place the JTAGC in reset through TAP controller state machine transitions controlled by TMS
2. Load the appropriate instruction for the test or action to be performed

Chapter 58
JTAG Data Communication (JDC)

58.1 Introduction

The JTAG Data Communication (JDC) module provides the capability to move register data between the IPS and JTAG domains. This facilitates communication between internal resources that access memory mapped register space and an external tool that accesses the JTAG port.

58.2 Overview

The JDC module consists of IPS accessible registers, JTAG accessible registers, and associated logic to coordinate movement of data from one register domain to another.

The JDC implements the following IPS data registers, occupying separate memory space:

- 32-bit memory mapped register that can be read or written via IPS (JOUT_IPS), and whose contents are ported out for capture into a JTAG register (JOUT) to be read via the JTAG port.
- 32-bit memory mapped register that can only be read via IPS (JIN_IPS), and whose contents are loaded from a JTAG register (JIN).

In addition to the data registers themselves, logic is implemented to indicate when new data has been shifted in via the JTAG port and is ready to be read from the JIN_IPS register, and when new data has been written to the JOUT_IPS register and is ready to be read via the JTAG port. The state of these flags is captured into a status register (MSR), and also provided to a JTAG register (JOUT) for tool visibility.

There is a single bus interface to port register data out to the JTAGC, and a single bus interface to port data in from the JTAGC. The architecture is shown in the following figure.

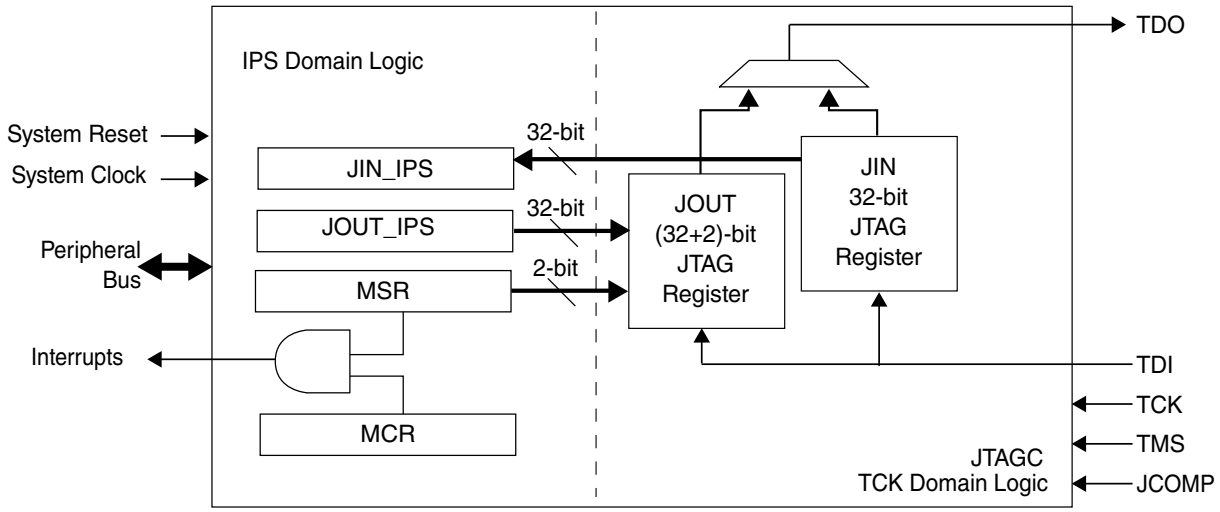


Figure 58-1. JDC block diagram

58.3 Memory mapped registers

Four 32-bit registers are implemented. Only 32-bit accesses are valid. The effects of access that are not 32 bits are not defined.

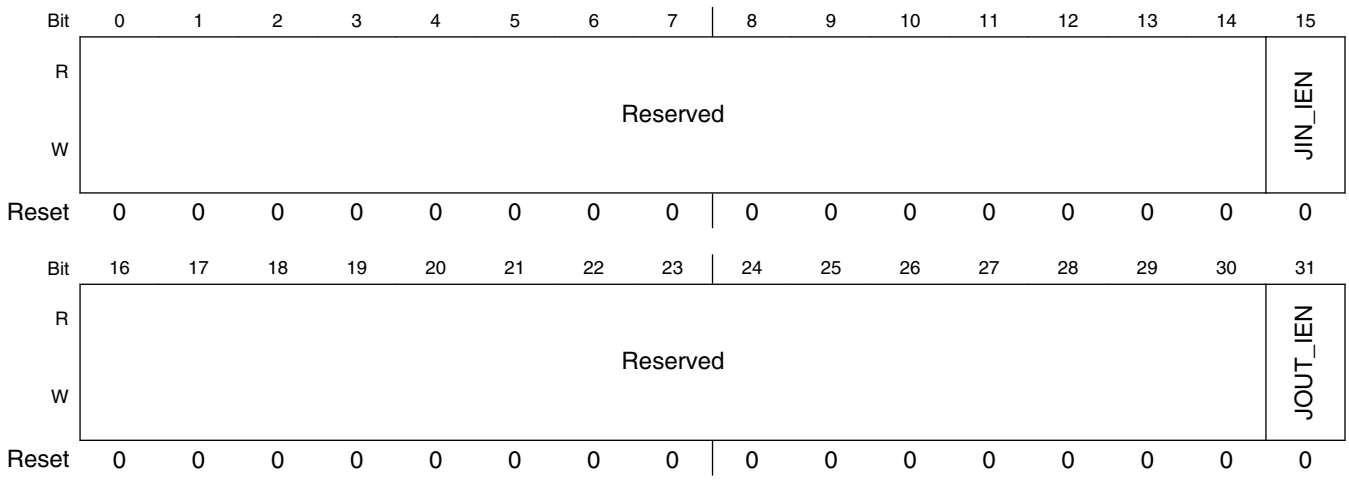
JDC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (JDC_MCR)	32	R/W	0000_0000h	58.3.1/2307
4	Module Status Register (JDC_MSR)	32	R/W	0000_0000h	58.3.2/2308
8	JTAG Output Data Register (JDC_JOUT_IPS)	32	R/W	0000_0000h	58.3.3/2309
C	JTAG Input Data Register (JDC_JIN_IPS)	32	R	0000_0000h	58.3.4/2310

58.3.1 Module Configuration Register (JDC_MCR)

The MCR is used to enable the interrupt outputs of the JDC. This register is reset by system destructive reset.

Address: 0h base + 0h offset = 0h



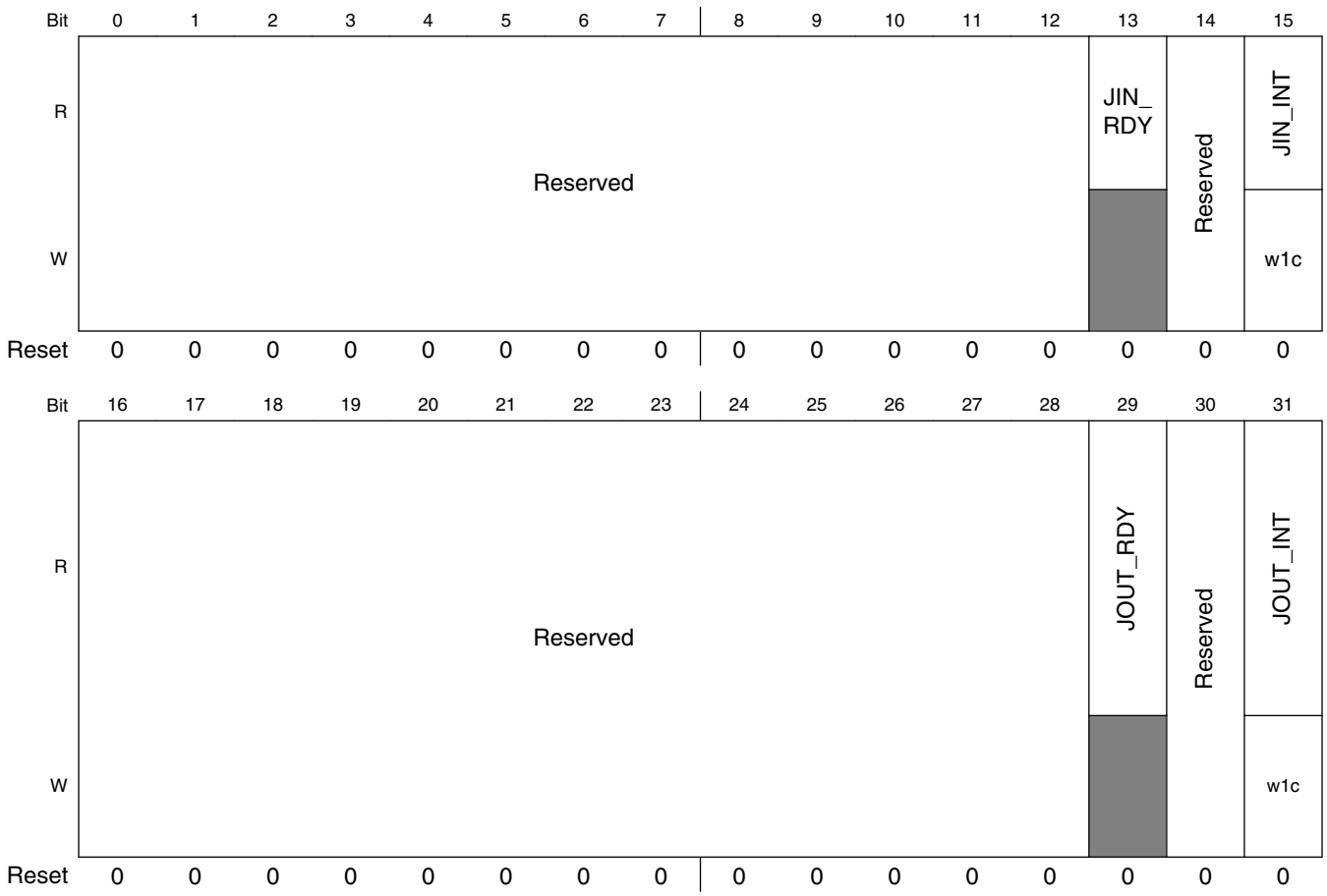
JDC_MCR field descriptions

Field	Description
0–14 Reserved	This field is reserved.
15 JIN_IEN	JIN Interrupt Enable. 0 Setting of MSR[JIN_INT] bit does not assert the JIN interrupt 1 Setting of MSR[JIN_INT] bit asserts the JIN interrupt
16–30 Reserved	This field is reserved.
31 JOUT_IEN	JOUT Interrupt Enable. 0 Setting of MSR[JOUT_INT] bit does not assert the JOUT interrupt 1 Setting of MSR[JOUT_INT] bit asserts the JOUT interrupt

58.3.2 Module Status Register (JDC_MSR)

The MSR holds the JTAG register status and interrupt bits. This register is reset by system destructive reset.

Address: 0h base + 4h offset = 4h



JDC_MSR field descriptions

Field	Description
0–12 Reserved	This field is reserved.
13 JIN_RDY	JIN Ready (read only). 0 Cleared upon software read of JIN_IPS contents via IPS 1 Set when new data is written to the JIN_IPS register
14 Reserved	This field is reserved.
15 JIN_INT	JIN Interrupt.

Table continues on the next page...

JDC_MSR field descriptions (continued)

Field	Description
	0 Cleared by writing logic 1 1 Set when new data is written to the JIN_IPS register
16–28 Reserved	This field is reserved.
29 JOUT_RDY	JOUT Ready (read only). 0 Cleared upon tool read of JOUT register via JTAG port 1 Set when new data is written to the JOUT_IPS register
30 Reserved	This field is reserved.
31 JOUT_INT	JOUT Interrupt. 0 Cleared by writing logic 1 1 Set when JOUT_RDY bit is cleared by tool reading JOUT register

58.3.3 JTAG Output Data Register (JDC_JOUT_IPS)

The JOUT_IPS register holds data written via IPS. The JOUT_IPS contents are ported out and captured into the JOUT register to be read via the JTAG port. This register is reset by system destructive reset.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Data																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JDC_JOUT_IPS field descriptions

Field	Description
0–31 Data	JOUT_IPS data.

58.3.4 JTAG Input Data Register (JDC_JIN_IPS)

Data written to the JTAG input data register (JIN) via the JTAG port is held in the JIN_IPS register, where it can be read via IPS. This register is reset by system destructive reset.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Data																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

JDC_JIN_IPS field descriptions

Field	Description
0–31 Data	JIN_IPS data.

58.4 Non-Memory mapped register definition

The JDC also implements two JTAG accessible registers that are not memory mapped. One JTAG register is used to shift in data to be placed in the JIN_IPS register. The other JTAG register captures data from the JOUT_IPS register and ready status from the MSR to be shifted out via the JTAG port.

58.4.1 JTAG output data register (JOUT)

The JOUT register captures data from the JOUT_IPS register upon execution of the JOUT_READ JTAG instruction. It also holds the JIN_RDY and JOUT_RDY status bits. This register is reset by system destructive reset and JTAG reset. The following figure shows the format of the JOUT register.

	33	…	2	1	0
R W	JOUT_IPS Data			JIN_RDY	JOUT_RDY
Reset	*[1]	*	*	*	*

1. The reset value of the JOUT register is 34'b0.

The JOUT register is described in the following table.

Table 58-1. JOUT JTAG register field descriptions

Field	Description
JOUT_IPS Data	Data value from JOUT_IPS register
JIN_RDY	State of JIN_RDY bit from MSR
JOUT_RDY	State of JOUT_RDY bit from MSR

58.4.2 JTAG input data register (JIN)

Data is written to the JIN register via JTAG during execution of the JIN_WRITE JTAG instruction. The JIN data is later captured in the JIN_IPS register to be read via IPS. This register is reset by system destructive reset and JTAG reset. The following figure shows the format of the JIN register.

	31	…	0
R W	JIN Data		
Reset	*[1]	*	*

1. The reset value of the JIN register is 32'b0.

The JIN register is described in the following table.

Table 58-2. JIN JTAG register field descriptions

Field	Description
JIN Data	Contains data to be captured in JIN_IPS register upon exit of Update-DR state when executing WRITE_JIN JTAG instruction.

58.5 Functional description

The JDC module provides the ability to shift in data via the JTAG port and capture that data into a memory mapped register that can be accessed via IPS. It also provides the ability to capture data written to a memory mapped register into a JTAG shift register for output via the JTAG port. An overview of the module functionality is described below.

- A software write to the JOUT_IPS register sets the JOUT_RDY flag bit, indicating new data is available to be read from the JOUT register via the JTAG port. The state of the JOUT_RDY bit is reflected in the MSR and also ported out to the JOUT register. A JTAG read of the JOUT register via execution of the JOUT_READ

instruction with a JOUT_RDY bit whose value is logic 1 indicates the register contains new data. The JOUT_RDY flag bit is cleared upon exit of the Capture-DR JTAG state during execution of the JOUT_READ instruction. Clearing the JOUT_RDY bit indicates to software that a new data value can be written to the JOUT_IPS register.

- A JTAG write to the JIN register via execution of the JIN_READ JTAG instruction updates the contents of the JIN_IPS register upon exit of the Update-DR state. An update of the JIN_IPS register sets the JIN_RDY flag bit, indicating new data is available to be read via IPS. The state of the JIN_RDY bit is reflected in the MSR register and also ported out to the JOUT register. The JIN_RDY flag bit is cleared upon software read of the JIN_IPS register. A JTAG read of the JOUT register with a JIN_RDY value of logic 0 indicates new data can be written to the JIN register.

58.5.1 JTAG register access

Refer to the JTAGC documentation for information on how to access the JTAG registers.

58.5.1.1 JDC block instructions

The JDC block implements the IEEE 1149.1-2001 defined instructions listed in the following table. This section gives an overview of each instruction; refer to the IEEE 1149.1-2001 standard for more details. All undefined opcodes are reserved.

Table 58-3. JTAG instructions

Instruction	Code[4:0]	Instruction Summary
Reserved	00001	Factory debug reserved
JOUT_READ	00010	Selects JOUT data register. Data from JOUT_IPS is captured into JOUT data register upon entry to Capture-DR state while JOUT_READ is active.
JIN_WRITE	01110	Selects JIN data register. Data from JIN is captured into JIN_IPS upon exit of Update-DR state while JIN_WRITE is active.
BYPASS	11111	Selects bypass register for data operations
Reserved	All other opcodes	Decoded to select bypass register

58.6 Use case example

A summary of how the JDC can be used to interact with the Hardware Security Module (HSM) to perform challenge/response password authorization is as follows:

1. After exit of reset, all registers are at their reset values. JIN_RDY bit value of 0 indicates the tool may write data to the JIN register. JOUT_RDY bit value of 0 indicates that software may write data to the JOUT_IPS register.
2. The MCR register is programmed to enable JOUT and JIN interrupts, or not.
3. Once the tool is ready to start the authorization process, it writes a value to the JIN register via execution of the JTAGC JIN_WRITE instruction.
4. Software monitors the state of the JIN_RDY bit or enables the JIN interrupt and uses the interrupt assertion as a trigger to start the authorization process.
5. Check if a Flash program/erase operation is in progress. If a Flash program/erase operation is in progress, wait for it to complete.
6. Once the authorization process is started, software writes the first JOUT value to the JOUT_IPS register. This sets the JOUT_RDY bit value to 1.
7. Following the initial tool write to the JIN register, the tool polls the JOUT register to determine when software has provided a challenge word. Upon reading the JOUT register with JOUT_RDY bit set, the tool records the data value to be used to generate the appropriate response. The read of the JOUT register clears the JOUT_RDY bit. The clearing of the JOUT_RDY bit also sets the JOUT_INT bit and asserts the JOUT interrupt if the MCR[JOUT_IEN] is set.
8. With the JOUT_RDY bit cleared, software may now provide the next challenge word.
9. Once the tool has generated a response word, it uses the value of the JIN_RDY bit from the JOUT register read to determine when it can perform a write to the JIN register to provide the response.
10. The HSM can provide additional challenge words and read back the corresponding response words as needed, repeating the process.

Simply providing a password without using the challenge/response protocol can be done in a similar way. The tool provides 32-bits of the password at a time with each write to the JIN register, and monitors the JIN_RDY bit value of the JOUT register to determine when the next JIN register value can be written. There is no need for software to write the JOUT_IPS register in this case.

Chapter 59
IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG)

59.1 References

- IEEE Std 1149.1-2001 (R2008), Standard Test Access Port and Boundary-Scan Architecture
- IEEE Std 1149.7, Standard for Reduced-Pin and Enhanced-Functionality Test Access Port and Boundary-Scan Architecture

59.2 Abbreviations

Table 59-1. Abbreviations

Abbreviation	Expansion
APU	Advanced Protocol Unit
APFC	Auxiliary Pin Function Control
CGM	Conditional Group Member
CGMC	Conditional Group Membership Count
CID	Controller ID
CJTAG	Compact JTAG
CLTAPC	Chip-Level TAPC
CP	Check Packet
CPA	Check Process Active
DCU	Data Channel Unit
EOT	End of Transfer
EPU	Extended Protocol Unit
Jscan	JTAG Scan
Mscan	Maximum-flexibility Scan
nTRST	Active low Test Reset
Oscan	Optimized Scan
PSS	Pause Selection State

Table continues on the next page...

Table 59-1. Abbreviations (continued)

Abbreviation	Expansion
RSU	Reset and Selection Unit
SP	Scan Packet
SPA	Scan Packet Active
Sscan	Segmented Scan
SSD	Scan Selection Directive
SSM	Scan State Machine
STL	System Test Logic
TAP	Test Access Port
TAP.1	An IEEE 1149.1 Test Access Port
TAP.7	An IEEE 1149.7 Test Access Port
TCA	Tap Controller Address
TCK	Test Clock
TCKC	Test Clock Compact
TDI	Test Data Input
TDIC	Test Data Input Compact
TDO	Test Data Output
TDOC	Test Data Output Compact
TMS	Test Mode Select
TMSC	Test Mode Select Compact
TS	Target System
ZBS	Zero Bit Scan

59.3 Introduction

The CJTAG module implements parts of the IEEE 1149.7 standard for test and debug capabilities. IEEE 1149.7 defines enhanced test and debug capabilities which are backwards compatible with IEEE P1149.1. This standard is also known as "Compact JTAG" or "CJTAG".

The module implements the following parts of the IEEE 1149.7 standard:

- IEEE 1149.1 compliant behavior while allowing multiple on-chip TAP controllers
- Ability to assert test reset
- Improved system scan performance with one-bit IR and DR chip bypass paths

- Physical connection of a Debug and Test System (external tool) to a running system without corrupting its operation
- Operation with both 4-pin and 2-pin scan topologies

59.3.1 Types of operation

The 1149.7 standard provides a scalable and flexible solution to meet the needs of varied component and system complexities. There are three types of operation:

- **Compliant**
  - Compliant with the IEEE 1149.1 standard
  - Four-pin operation using the "Standard Protocol" — IEEE 1149.1 signaling where TCK, TMS, TDI and TDO signals transfer control/data information
  - More than one TAPC may be associated with the TAP (once made visible by private instructions)
- **Extended**
  - Compatible with the IEEE 1149.1 standard
  - Four-pin operation using the "Standard Protocol" — IEEE 1149.1 signaling where TCK, TMS, TDI(C) and TDO(C) signals transfer control/data information
  - Additional test/debug functions
- **Advanced**
  - Two-pin operation using the "Advanced Protocol" — signaling where TCK(C) and TMS(C) signals transfer control/data information
  - The TMS(C) signal is operated in a bidirectional manner with the TMS, TDI and TDO TAP information serialized and transferred via this signal

59.3.2 Deployment by class

The TAP.7 capabilities are deployed using five capability classes (T0–T4), each matching the needs of certain scan topologies and mixes of 1149.1 TAPs and 1149.7 TAPs. The classes provide the following capability:

- **Compliant Operation** – provides behavior compliant with the IEEE 1149.1-2001 standard:
  - **Class T0** – 1149.1 compliance when multiple TAPs are used on the same chip.
- **Extended Operation** – extends the capabilities of the IEEE 1149.1-2001 standard:
  - **Class T1** – T0 + Test access port power management, functional reset, and test reset.
  - **Class T2** – T1 + Chip bypass (1-bit) for IR/DR Scans + hot-connection with no errors..
  - **Class T3** – T2 + Star operation (TAPs are paralleled) with series equivalent scans for test.
- **Advanced Operation** – adds capabilities addressing test and debug needs in complex environments:
  - **Class T4** – T3 + Scan with two pins with the Advanced Protocol. A number of protocol optimizations maximize scan performance. TDI(C)/TDO(C) pins are optional and may have programmable function when implemented.

59.3.3 1149.7 TAP signals

The TAP signals for the T0–T4 TAP.7 classes are listed in the following table. Signal names that end in “C” are associated with expanded functionality (beyond the functionality provided by the IEEE 1149.1 standard).

The table uses the following notation:

- M = Mandatory signal
- MC = Mandatory signal with expanded functionality
- O = Optional signal
- OC = Optional signal with expanded functionality

The CJTAG module provides T4 capability and implements the following pins:

- TCKC
- TDIC
- TDOC
- TMSC

Table 59-2. TAP.7 signal list

Signal name	Description	Class				
		T4	T3	T2	T1	T0
TCK/TCKC	Test clock	MC	M	M	M	M
TMS/TMSC	Test mode select	MC	M	M	M	M
TDI/TDIC	Test data input	OC	MC	M	M	M
TDO/TDOC	Test data output	OC	MC	M	M	M
Mandatory signal count (minimum)		2	4	4	4	4

59.3.4 TAP.7 architecture

The TAP.7 hardware architecture is shown in the following figure. It is described with the hardware layers listed below:

- **STL – System Test Logic** – Logic with 1149.1 compliant behavior and underlying TAP hierarchy (T0 capabilities). Provides an IEEE 1149.1 interface for the T0 TAP. 7. This logic is outside the CJTAG module.
- **RSU – Reset and Selection Unit** – A hardware layer that is placed between the APU, EPU, or STL and the TAP.7 signals (Added as an option to support the use of the Control Protocol) Provides reset and TAP.7 Controller selection services.
- **EPU – Extended Protocol Unit** – A hardware layer that is placed between the STL and the TAP.7 signals. (Added for T1, T2, and T3 capabilities). Provides an IEEE 1149.1 interface for the T1–T3 TAP.7s.
- **APU – Advanced Protocol Unit** – A hardware layer that is placed between the STL/EPU and the TAP.7 signals (added for T4 capabilities). Provides a T4 TAP.7 interface that is either narrow or wide, with the wide version providing an IEEE 1149.1 interface.

The CJTAG module supports T4 functionality that includes STL, RSU, EPU, and APU hardware layers.

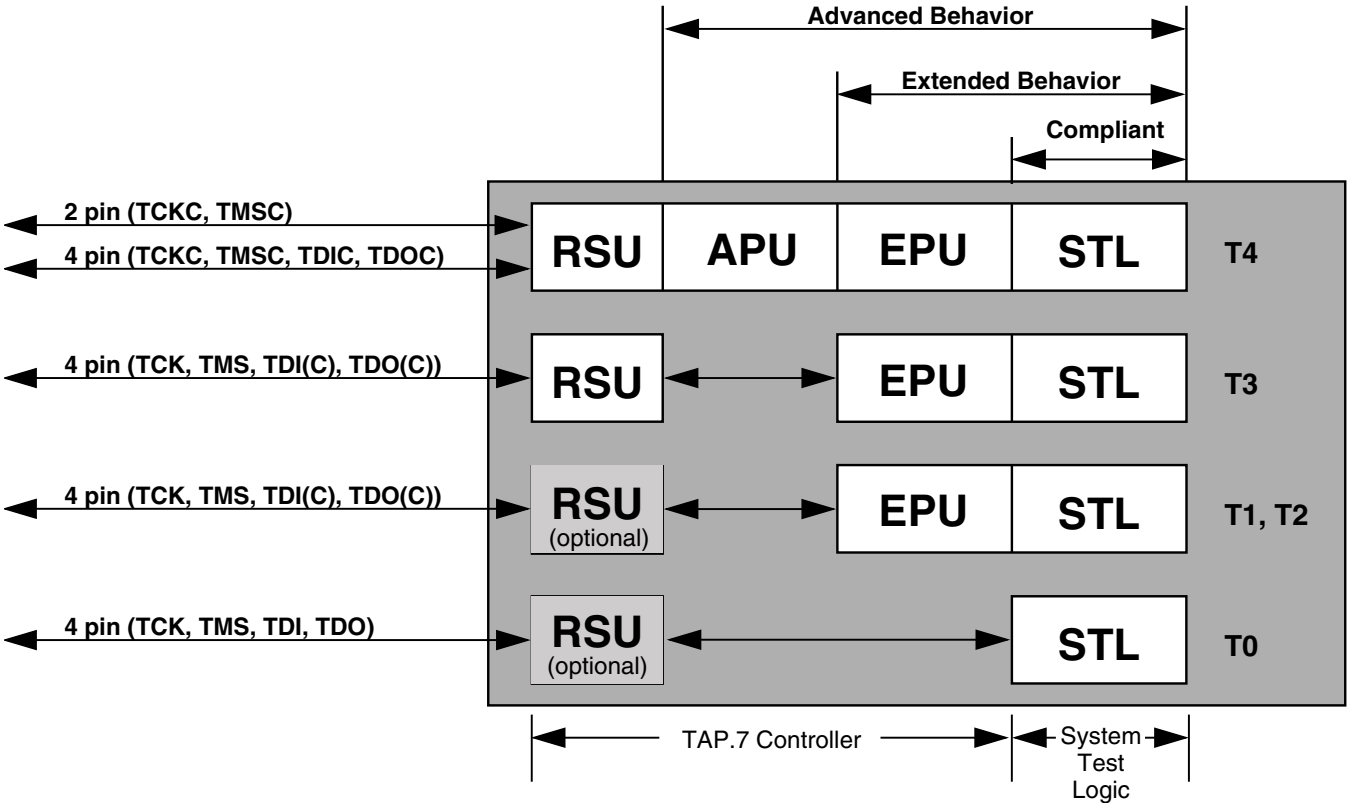


Figure 59-1. TAP.7 Controller Architecture

59.3.5 Protocols

The TAP.7 architecture utilizes the three protocols shown in the following figure.

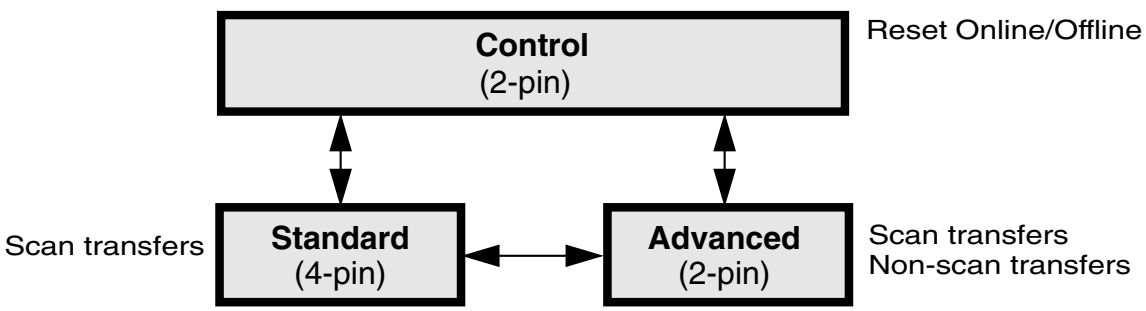


Figure 59-2. TAP.7 Behaviors

The Standard Protocol transfers data with the TMS(C) signal value being sampled with a TCK(C) edge. The Advanced and Control Protocols use both data transferred with the TMS(C) signal value being sampled with a TCK(C) signal edge and escape sequences. The Standard Protocol may be used to switch to the use of the Advanced Protocol and

vice versa. The Control Protocol can also be used to switch between the use of the Standard and the Advanced Protocol. The mandatory and optional deployment of these protocols with the TAP.7 classes is shown in the following table.

Table 59-3. Class/protocol relationship

Class	Protocol		
	Standard	Control	Advanced
T0 – T2	Mandatory	Optional	N/A
T3	Mandatory	Mandatory	N/A
T4	Mandatory	Mandatory	Mandatory

The CJTAG module implements T4 functionality and supports the Standard, Advanced, and Control Protocols.

59.3.5.1 Standard Protocol

The Standard Protocol is the signaling defined by the IEEE 1149.1 standard. This protocol is also used to implement TAP.7 commands using only the TCK(C) and TMS(C) signals in a manner that does not require either TAP.7 controller instruction or data registers. The TAP.7 controller does not add bits to the scan paths of underlying technology. The Standard Protocol can be used to manage the TAP.7 controller functionality independent of the function associated with the availability of the TDI and TDO signals. The Standard Protocol supports STL data transfers only when these signals are present and provide the 1149.1 functionality defined by the IEEE 1149.1 standard.

59.3.5.2 Advanced Protocol

The Advanced Protocol supports both STL data transfers and management of the TAP.7 controller functionality with both the 2-signal (narrow) and 4-signal (wide) configurations. The Advanced Protocol serializes the TMS, TDI, and TDO information used with the Standard Protocol using only the TCK(C) and TMS(C) signals. A number of additional scan formats provide bit sequences optimized for specific use cases. Each of these additional scan formats affects the TMSC signaling sequences.

59.3.5.3 Control Protocol

The Control Protocol uses Escape sequences to reset the TAP.7 controller, place a TAP.7 controller offline or online, and indicate End of Transmission (EOT) with the Advanced Protocol. The characteristics of this protocol provide Hot-Connect-Protection by requiring a preamble of alternating ones and zeroes preceding a selection escape when a TAP controller is started up offline.

59.4 Operating models

The following figure illustrates a simplified view of the CJTAG operating model.

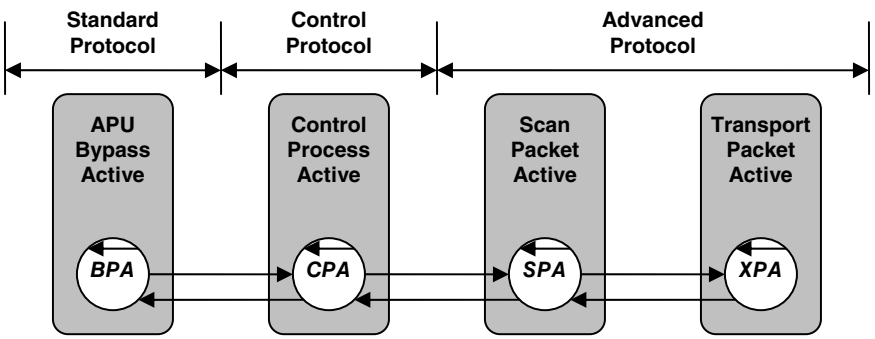


Figure 59-3. CJTAG operating model

59.5 CJTAG implementation summary

The CJTAG module is a T4 TAP.7 controller that implements the following functions.

59.5.1 T0 functions

- IDCODE. - implemented in chip level TAPC (JTAGC module)
- 1149.1 Compliance at start-up
- Isolating / Excluding of embedded TAPs – implemented in chip level TAPC (JTAGC module)
- Operation with other technologies and scan topologies
- Multiple embedded TAPs

59.5.2 T1 functions

- EPU Command processing
- EPU Class-Specific Registers
- Asserting a test reset to the STL (TRESET Register)

59.5.3 T2 functions

- "Super Bypass" function – JScan0, JScan1, JScan2 scan formats. Allows the STL to be coupled or decoupled.

59.5.4 T3 functions

- Generation of a TAP.7 controller reset with a Reset Escape Sequence
- JScan3 Scan Format

59.5.5 T4 functions

- TMSC sampling on rising edge or falling edge of TCKC
- MScan (For test and debug)
- OScan 0-7
- TDIC and TDOC / Alternate pin function multiplexing (wide 4 only)

59.6 Ancillary services

59.6.1 Overview

Ancillary services are pertinent to any TAP.7 controller implemented with an RSU, such as the CJTAG module.

Ancillary services include:

- Resets
- Start-up options

- Escape sequences
- TAPC state machine

59.6.2 Resets

The TAP.7 controller reset function expands the reset function provided by the IEEE 1149.1 standard to provide additional reset types. It supports six reset types, Type-0 – Type-5, listed below. It can be implemented with as few as two reset types (Type-2 and Type-4), or as many as six reset types (all types) of reset inputs: This is determined by the class and options implemented.

The CJTAG module utilizes Type-0, Type-2, Type-3, Type-4, and Type-5 resets.

- **Type-0** – Reset of the TAP.7 by power management logic (power-on reset)
- **Type-1** – Chip level generated Test Reset. (equivalent to Type-2)
- **Type-2** – Externally generated Test Reset (nTRST/nTRST_PD pin)
- **Type-3** – TAP.7 controller generated reset
- **Type-4** – *Test-Logic-Reset* state
- **Type-5** – TAP.7 Controller TReset Register reset of the CLTAPC.

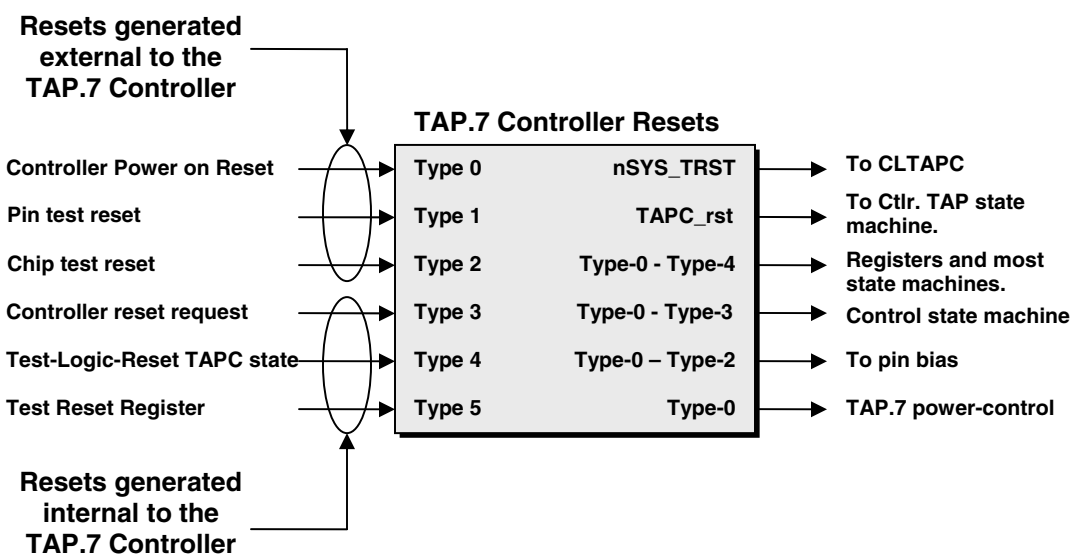


Figure 59-4. Conceptual view of the EPU reset logic and state machines

59.6.2.1 Type 5 reset

A Type-5 reset affects only the STL via the nSYS_TRST pin. It is asserted when the TRESET Register value is a logic 1. This register is set to its inactive state by reset Types 0-4.

59.6.2.2 Type 4 reset

A Type-4 reset is created by the *Test-Logic-Reset* state. It does not affect the TAP.7 TAPC state machine state as this state is the source of this reset, nor does it cause the assertion of the nSYS_TRST signal. It does however affect the coupling and decoupling of the STL. The Type-4 reset initializes the TAP.7 registers and most controller state machines. It affects but does not initialize the control state machine managing TAP.7 controller selection. It also establishes the default operating conditions.

59.6.2.3 Type 3 reset

A Type-3 reset is created by reset requests generated by the TAP.7 controller. A Type-3 reset performs the functions associated with a Type-4 reset and creates the *Test-Logic-Reset* state.

A Type-3 reset is generated in response to reset requests generated by EPU and APU logic

- A reset escape sequence
- Reset protocol sequences (delay packet directive, check packet directive)

A Type-3 reset request is generated with a reset escape sequence. This function is similar to but not the same as the reset function generated by a test reset pin. The Advanced Protocol includes certain bit patterns that initiate Type-3 resets. These bit patterns assist in the initialization of the TAP.7 controller when the TCKC signal is sourced by the TS and the external tool/TS connection is broken. The external tool may also generate these sequences, if desired, while the external tool/TS connection remains intact. These reset requests cause a Type-3 reset that is one clock wide.

With this being the case, consideration may be given to integrating the CJTAG module without the use of the Test Reset signal (nTRST/nTRST_PD pin).

59.6.2.4 Type 2 reset

A Type-2 reset is created by chip level logic to initialize the TAP.7 controller when TAP.7 power control is not implemented. It is unimplemented otherwise. It performs all of the functions of the Type-3 reset in addition to initializing the escape sequence detection logic.

59.6.2.5 Type 1 reset

A Type-1 reset is created with either the nTRST or nTRST_PD pin. It is not created when neither of these pins is implemented. It performs all of the functions of the Type-2 reset. It initiates TAP.7 controller power down provided the TAP.7 power control mode permits power down.

59.6.2.6 Type 0 reset

A Type-0 reset is implemented when TAP.7 controller power down is implemented. It performs all of the functions of a Type-1 reset and also initializes the TAP.7 power control logic.

59.6.2.7 Reset State Machine

The Reset State Machine assists in the translation of the reset inputs to the reset outputs. It handles Type-3 reset controller reset requests. The state machine state assures Type-0, Type-1, and Type-2 resets are forwarded to the STL without modifying their duration.

This state machine allows a Type-3 reset only when its state is “T3 Reset” (two TCK(C) falling edges have occurred in the absence of a Type-0, Type-1, and Type-2 reset).

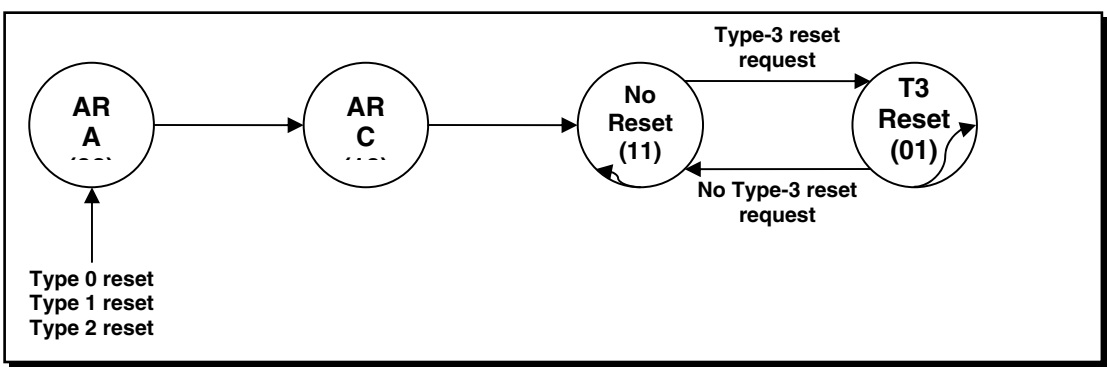


Figure 59-5. Reset state machine

Table 59-4. Reset state descriptions

Reset state		Description
ARA	Async. Reset Asserted	An async. reset has occurred or is occurring
ARC	Async. Reset Completed	An async. reset has occurred and has been released
No Reset	No Reset	No Reset
T3 Reset	Type-3 Reset	A Type-3 reset request has been processed.

Type-0, Type-1, and Type-2 resets are fully asynchronous. These resets are passed directly to nSYS_TRST without modification. The Type-3, Type-4, and Type-5 resets are fully synchronous.

The effects of the TAP.7 controller reset types are shown in the following table.

Table 59-5. Reset effects

Reset	CLTAPC	Pin-Hi-Z	State Machines other than CSM	Control State Machine (CSM)	TAPC SM State	Escape Sequence Detection	Power Control State
Type-5	Yes	–	–	–	–	–	–
Type-4	–	Yes	Yes	–	–	–	–
Type-3	Yes	Yes	Yes	Yes	Yes	–	–
Type-2	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-1	Yes	Yes	Yes	Yes	Yes	Yes	–
Type-0	Yes	Yes	Yes	Yes	Yes	Yes	Yes

59.6.3 Start-up Options

59.6.3.1 Overview

The TAP.7 start-up options are shown in the following table. The only startup option currently supported by the CJTAG module is 1149.1 Compliant.

Table 59-6. Start-up Options

Start-up Option	Description
1149.1 Compliant	TAP.1 signal functionality, the CLTAPC is coupled
1149.1 Compatible	TAP.7 signal functionality, the CLTAPC is decoupled

Table continues on the next page...

Table 59-6. Start-up Options (continued)

Start-up Option	Description
1149.1 Protocol Compatible	No, or alternate TDIC/TDOC signal functionality, the CLTAPC is decoupled.
Offline-at-Start-up (Dormant)	The TAP.7 controller is offline and awaits synchronization to external tool operation so as to be placed online, the CLTAPC is decoupled after its state is moved to *Run-Test/Idle*

59.6.3.2 1149.1 Compliant start-up

Start-up behavior with the 1149.1 Compliant behavior start-up option:

- 1149.1-specific behavior / Standard protocol
- The TAPC finite state machine operated in lock step with the CLTAPC finite state machine.
- The TDIC and TDOC pins are present and provide the TDI and TDO function.
- The CLTAPC is coupled at start-up.

With this start-up option, the TAP.7 controller may be programmed and the scan topology may be interrogated. If the scan topology is known to be the Series Scan Topology at power-up, the TAP.7 can be operated as though it is a TAP.1.

Table 59-7. Reset values for TAP.7 startup options

Start-up Option	Type-0 – Type-3 Reset?	Default TAP.7 and TAP.7 controller characteristics				
		TAP.7 Controller online	CLTAPC Coupled	SGC Reg. == 1	SCNFMT	Initial TMSC pin bias
1149.1 Compliant	x	Yes	Yes	Yes	JScan0	PU

59.6.4 RSU operation

The RSU uses TCKC and TMSC with the control protocol for configuration control.

59.6.4.1 General Operation

From a very high level, the RSU enables the remainder of the TAP.7 functionality when the TAP.7 controller is Online and disables the remainder of the TAP.7 functionality otherwise. The RSU either consumes or discards the TMS(C) information while the TAP. 7 controller is Offline. The relationship of the escape sequence detection, technology selection logic, and the underlying technology is shown in the following figure.

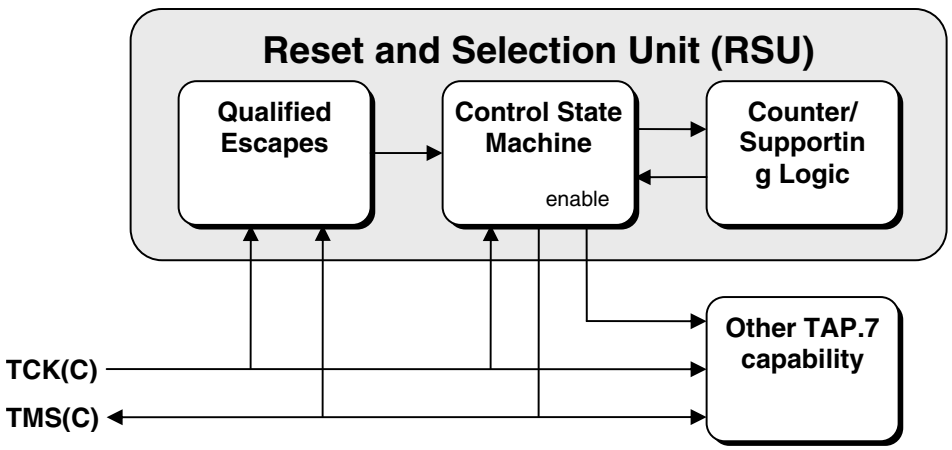


Figure 59-6. Conceptual view of technology selection mechanism

59.6.4.2 Common Signaling Across Technologies

With the TAP.7 architecture, the signaling conventions chosen for technology selection is multiple data transitions of the TMS(C) signal while the TCK(C) signal is a logic 1. The common signaling space is available when the external tool sources the TCK(C) signal as it is capable of creating these signaling conventions. It is unavailable when the TS sources the TCK(C) signal as the external tool cannot stop the TCKC signal at a logic 1 to create these signaling conventions.

59.6.4.3 Escape Sequence Detection

TAP.7 TAPC interprets the count of the number of TMS(C) edges while TCK(C) is a logic 1 as one of four escape sequences. Each escape sequence has a different function and TMSC edge count. These escape sequences, their edge counts, and their functions are described below:

- **Custom** (2 or 3 edges) – Ends scan and transport transfers with a TAP.7 controller, may be used for other purposes with another technology.
- **Reset** (> 7 edges) – Resets all technologies (generates a Type-3 TAP.7 TAPC reset)

The external tool creates an escape sequence by generating one or more TMS(C) edge pairs while the TCK(C) signal is a logic 1 value. Even and odd TMS(C) edge counts beginning with two are given the same meaning.

An escape sequence:

- Can be detected only when a Type-0, Type-1 or Type-2 reset is **not** asserted.
- Begins and ends while the TCK(C) is a logic 1.
- Uses TMS(C) as a clock while the TCK(C) is a logic 1.
- Overlays additional control information onto the normal information that is transferred with the TCK(C) and the TMS(C) without changing the normal information.
- Must be detected when TCK(C) is a logic 1 regardless of the TCK(C) and TMS(C) drive histories.

59.6.4.3.1 Custom escape sequence

An online technology can use the custom escape sequence in any manner it chooses. It is used as the End of Transmission (EOT) of certain data types (formats OScan4-7) with T4 and above TAP.7s.

59.6.4.3.2 Reset escape sequences

A reset event initializes all technologies sharing the TCK(C) and TMS(C) connectivity. A reset escape sequence resets the technology. It is recommended that this reset occur with the falling edge of TCKC. With a TAP.7 controller a reset escape sequence generates a Type-3 reset beginning with the falling edge of TCK(C) following the asynchronous detection of the event.

59.6.5 TAPC State Machine

T1 and above TAP.7s require knowledge of the TAPC state machine state. With T2 and above TAPs this knowledge must be developed independently of the CLTAPC. This is generally accomplished by implementing a TAPC state machine within the EPU. This TAPC state machine provides the function of the state machine within a standard 1149.1 TAPC. It is recommended that this state machine is implemented to change state on the falling edge of TCK(C) as this has numerous advantages to a TAP.7 controller. The EPU does not contain 1149.1 Instruction or Data Registers. The APU controls the state changes of this TAPC state machine with T4 and above TAP.7s.

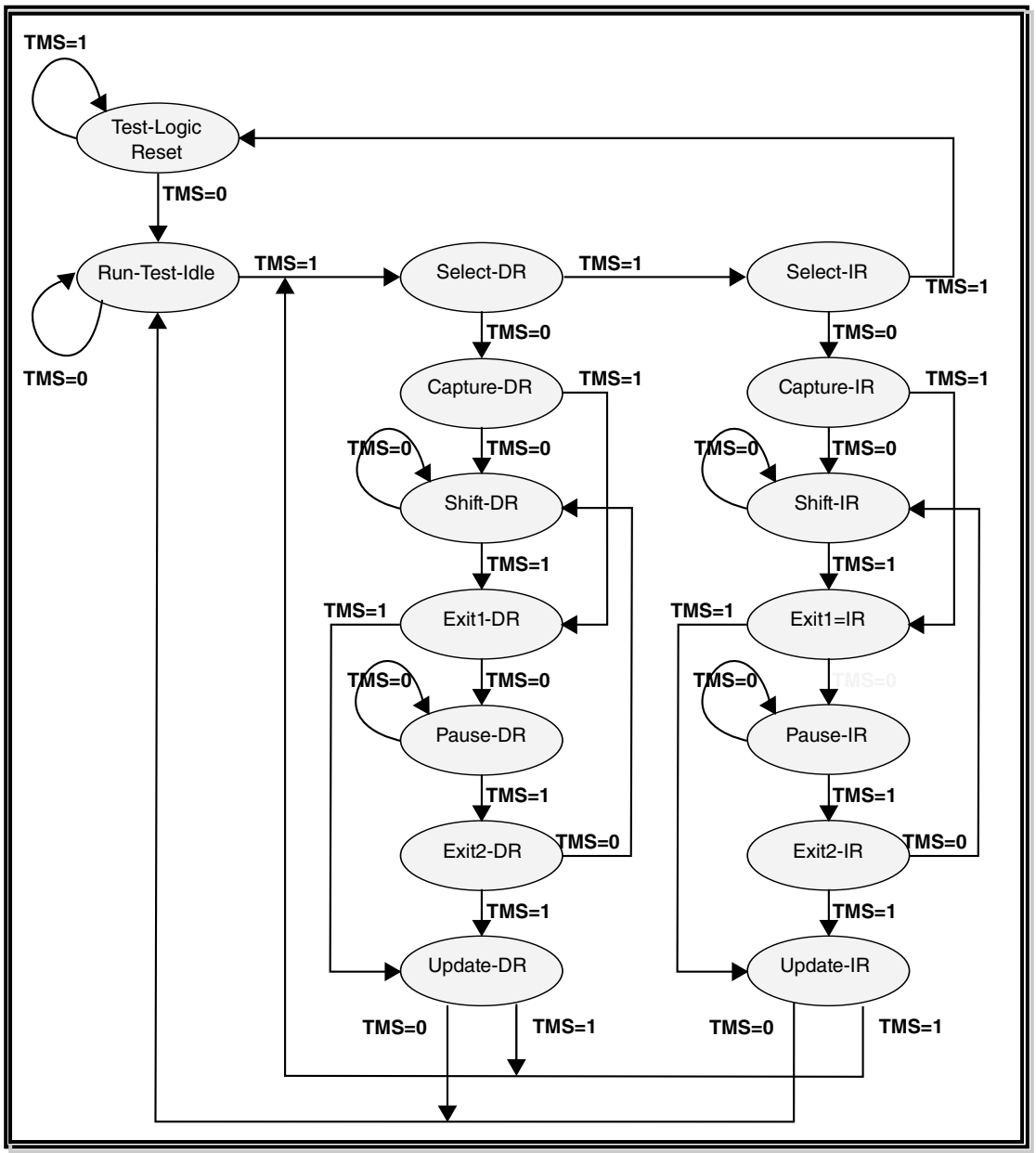


Figure 59-7. Falling edge TAP controller state machine

59.7 EPU (Extended Protocol Unit) Operation

59.7.1 EPU Operation

The EPU adds functionality to the BYPASS and IDCODE instructions.

59.7.1.1 Zero-bit DR-Scans (ZBS)

A zero-bit DR-Scan (ZBS) is a TAPC-state sequence that begins with the *Select-DR-Scan* state and ends with an exit from the *Update-DR* state without an intervening *Shift-DR* state. The external tool controls the EPU using ZBSs.

The use of ZBSs can begin immediately following the *Test-Logic-Reset* state since this state initializes the Instruction Registers of TAPs with either the *BYPASS* or *IDCODE* instructions.

59.7.1.2 Utilizing ZBSs for TAP.7 Controller Functionality

Beginning with a ZBS count of zero, the ZBS count is incremented with each consecutive occurrence of a ZBS without encountering a *Shift-DR* state. This count cannot be incremented past seven. *Run-Test/Idle* is the only state that may occur between consecutive occurrences of ZBSs without terminating the count of consecutive ZBSs.

59.7.1.3 Locking the ZBS Count

When a DR-Scan containing a *Shift-DR* state occurs, and the ZBS count is greater than zero, the ZBS count is not incremented when the *Update-DR* state is reached. It is instead locked at its current value. ZBSs occurring after locking the ZBS count do not affect the locked ZBS count. Locking the ZBS count is the equivalent of storing the count for subsequent use.

59.7.1.4 Control levels

Locking the ZBS count activates a control level that is equal to the locked ZBS count (1-7) when ZBSs are being used for TAP.7 functionality and merely locks the ZBS count otherwise.

1. ZBSs may be used by the STL or another function
2. TAP.7 command generation
3. Reserved control level
4. Accesses to optional TAP.7 scan paths
5. Accesses to optional TAP.7 scan paths

6. Force offline operation (optional)
7. External tool uses

Control level one may be used by the STL since there is no TAP.7 controller functionality associated with this control level.

Control level two enables DR-Scans to create TAP.7 commands. A portfolio of these commands supports the deployment of TAP.7 controllers in the Series or Star Scan Topologies. These commands manage TAP.7 controller registers and perform other functions.

Control level three is reserved for use with a subsequent specification revision.

Control levels four and five provide access to custom (private) scan paths when access to these paths is permitted by a TAP.7 register bit used for this purpose. The EPU provides a 1-bit scan path otherwise. No other EPU functionality is associated with these control levels. The CJTAG module does not support the use of this control level.

Control level six may be optionally used to force offline operation with a TAP.7 controller that supports this capability. The EPU provides a 1-bit scan path for this control level. The CJTAG module does not support the use of this control level.

Control level seven is reserved for external tool use. With this control level, external tool capabilities may be managed using the same infrastructure used to manage TAP.7 controller capability. The EPU provides a 1-bit scan path for this control levels.

59.7.1.5 Exiting a Control Level

A control level is exited when one of the following occurs:

- The *Select-IR-Scan* TAP.7 controller state
- The *Test-Logic-Reset* state
- Certain TAP.7 controller commands (Exit Control Level – ECL register bit)
- An event that synchronizes the operation of TAP.7 controllers

59.7.1.6 Zeroing the ZBS Count

Events causing an exit from a control level also zero the ZBS count. Scan Selection Directives (SSDs) associated with T3 and above TAP.7s zero the ZBS count when the control level has not been locked.

59.7.1.7 Utilizing ZBSs within the STL

After the *Test-Logic-Reset* state, ZBSs are used for basic EPU control. ZBSs that are intended for TAP.7 use are henceforth called "***TAP.7 ZBSs***". ZBSs that are intended for other uses are henceforth called "***Other ZBSs***". Two TAP.7 registers enable the STL's use of ZBSs. Storing these registers causes a control level exit, zeroes the ZBS count, and enables the use of Other ZBSs. Storing the Suspend Register (**SUSPEND**) suspends the use of control levels. Storing the ZBS Inhibit Register (**ZBSINH**) inhibits ZBS detection.

When either of these methods is used, the *Test-Logic-Reset* state enables the use of TAP. 7 ZBSs in addition to initializing the TAP.7 controller.

The operation of the TAP.7 controller changes with the use of these two paths.

When the System Path is used:

- A count of eight consecutive ZBSs is used to create a logic 0 SUSPEND Register value.
- The CLTAPC supplies the TAP.7 scan path and controls the drive of TDO(C), provided it is not bypasses.
- The EPU supplies the TAP.7 scan path (a one bit bypass) when the CLTAPC is bypassed.
- Functions supporting Boundary Scan in a Star Scan Topology are enabled.

When the EPU Path is used:

- When the control level has not been locked, the TDO(C) signal remains high-impedance.
- The command control level may be created
- When commands are used, the TDO(C) signal remains high-impedance except during *Shift-DR* states within CR-Scans, and remains high impedance at other times.
- Auxiliary scan paths may be accessed with control levels four and five.

59.7.2 EPU Registers

59.7.2.1 Global

A global register is a register whose value is stored at the same time and with the same value as registers with the same name in all TAP.7 TAPCs sharing an external tool connection. They are stored using TAP.7 TAPC commands dedicated for this purpose. These commands store a single global register. Global registers manage TAP.7 TAPC functions that affect the synchronized operation of a TAP.7 TAPC. An example of a global register is the Scan Format (SCN_FMT) Register used to define the TAPC signal protocol.

A second means of storing all global registers is available. The selection of the TAP.7 TAPC with the technology selection function stores either the Scan Format register value (short form) or all global registers (long form). Storing all global registers provides a simple means to synchronize the values of all global registers when Technology Selection causes a non-operating TAP.7 TAPC to become operational while other TAPCs are operational.

59.7.2.2 Local

Local registers manage TAP.7 TAPC functions that do not affect the synchronized operation of TAP.7 TAPCs. Local registers are conditionally stored using TAP.7 controller commands dedicated for this purpose. These commands provide for changing the value of these registers one a time with a unique value in a Series Scan Topology based on the TAPC's position on the scan chain. Local registers are stored with a unique value in a Star Scan Topology based on a TAPC address assigned to the TAPC.

Registers programmed with commands are described in this section. The CJTAG module implements all mandatory T4 registers as well as the optional APFC, TRESET, and RDBACK0 registers.There are 16 mandatory and 10 optional registers that can be grouped in classes as shown in Read-only – Configuration and register read back..

- **Control** – TAP.7 controller behaviors and characteristics.
- **Options** – Control of optional functions.
- **Select** – Controls the selection of a TAP.7 for Scan and execution of conditional commands.
- **Read-only** – Configuration and register read back.

Table 59-8. TAP.7 controller register list (managed with commands)

Register type	Global/local	Width	Register mnemonic	Name
Control	Global	1	ECL	Exit Control Level
		1	SUSPEND	Suspend
		1	ZBSINH	ZBS Detect Inhibit
		5	SCNFMT	Scan Format
		1	SSDE	SSD Enable
		2	DLYC	Delay control
		2	RDYC	Ready Control
Options	Local	1	TRESET	Test Reset
		2	APFC	Aux. Pin Func. Cntl.
Select	Local	1	CGM	Cond. Group Member
		1	SGC	Scan Group Candidate
		1	SEDGE	Sampling Edge
Read-only	Local	32	RDBACK0	Read-back 0
		32	RDBACK1	Read-back 1
		32	CNFG0	Configuration Register 0

M = Mandatory register

O = Optional register

59.7.2.3 Register Descriptions

59.7.2.3.1 Exit Control Level (ECL)

Storing this register zeroes the ZBS count and unlocks the control level.

This register is set using the STMC command. It remains logic 1 only momentarily before immediately returning to logic 0.

This register is cleared by a global register load.

59.7.2.3.2 Suspend (SUSPEND)

Setting this register suspends the use of control levels. When this register is a logic 1, the "System Path" (STL) is used.

1 – Suspend use of control levels. Enables "System Path."

0 – Enable use of control levels.

SUSPEND is set with the STMC command.

This register is cleared when 8 consecutive ZBSs are detected without the locking of the ZBS count.

This register is cleared by a global register load.

59.7.2.3.3 ZBS Inhibit (ZBSINH)

Storing to ZBSINH inhibits ZBS detection.

1 – Inhibit ZBS detection.

0 – Enable ZBS detection.

ZBSINH is set with the STMC command.

This register is cleared by a global register load.

59.7.2.3.4 Scan Format (SCNFMT[4:0])

Set using Store Scan Format (STFMT) command.

Read using Scan String (SCNS) command.

Table 59-9. Scan Format Register Values

TAPC.7 Class	SCNFMT	Description
T2–T4	00000	**JScan0:** • 1149.1 Compliant Operation. • Requests the coupling of the CLTAPC when the Run-Test/Idle state is reached.
	00001	**JScan1:** • 1149.1 Compatible Operation • Requests the decoupling of the CLTAPC when the Run-Test/Idle state is reached.
	00010	**JScan2:** • 1149.1 Compatible Operation. • Requests the coupling or decoupling of the CLTAPC when the Run-Test/Idle state is reached based on the value of the CACT register
T3–T4	00011	**JScan3:**

Table continues on the next page...

Table 59-9. Scan Format Register Values (continued)

TAPC.7 Class	SCNFMT	Description
		• Non-compatible 1149.1 Operation • Allows the detection of a T2 TAP.7 controller that is deployed in a Star-4 topology. • Enables TDOC signal behavior that prevents drive conflicts in a Star-4 Scan Topology. • Enables the use of SSDs
T4	01000	**OScan0** • Data rate dependent components • Single TAP.7 communication • Debug and Test applications
	01001	**OScan1** • TAP.1 components • Single/Multi-TAP.7 communication • Debug and Test applications
	01010	**OScan2** • TAP.1 components • No TDI or TDO in non-shift states • Debug applications
	01011	**OScan3** • TAP.1 components • No TDI in non-shift states • No TDO in any state
T4	01100	**OScan4** • Data rate dependent components • Single TAP.7 communication • Debug and Test applications
	01101	**OScan5** • TAP.1 components • Single/Multi-TAP.7 communication • Debug and Test applications
	01110	**OScan6** • TAP.1 components • No TDI or TDO in non-shift states • Debug applications

Table continues on the next page...

Table 59-9. Scan Format Register Values (continued)

TAPC.7 Class	SCNFMT	Description
	01111	**OScan7** • TAP.1 components • No TDI in non-shift states • No TDO in any state
	10000	**MScan** • Data rate dependent components • Multi TAP.7 communication • Directed CID assignment • Virtually any IP with compliant or non-compliant 1149.1 behavior
	10001 – 11111	**Reserved**

The use of a Scan Format value other than 00000b – 10000 shall not change the value of the SCNFMT Register.

59.7.2.3.5 Scan Selection Directive Enable (SSDE)

This register specifies whether the Scan Selection State may be managed using the scan selection directives.

0 – Scan selection directives are not enabled.

1 – Scan selection directives are enabled.

This register is set using the STMC command.

This register is read using SCNS (Scan String), RDBACK0.

59.7.2.3.6 Delay Control (DLYC[1:0])

The Delay Control register provides a means for the external tool to delay the completion of an SP. This delay is essentially an external tool stall. This delay may be zero, one, two, or n TCKC periods. This delay is especially useful when the TCKC is sourced by the TS and a Standard-to-Advanced Protocol adapter is added to an external tool supporting only the Standard Protocol. It allows an external tool design additional time to receive TDO, advance the external tool TAPC state, receive TMS and TDI information, and begin generation of a new SP payload. The DLYC register should be programmed before the use of the Advanced Protocol begins after the use of the Standard Protocol.

Set using the STMC command.

Read using SCNS, RDBACK0

Bits 18-17 of the "Global Register."

59.7.2.3.7 Ready Control (RDYC[1:0])

The Ready Control register defines behavior exhibited in the SP payload output bit frames when the STL stall opportunities (RDY bits) are included as control information. This register defines the number of additional bits inserted in these bit frames. These bits provide more time for a high-performance external tool to ascertain whether the TAP.7 controller is ready to complete the SP payload. The use of these bits makes a high-performance external tool design easier as input and output buffer delays play less of a role in limiting the TAP.7 controller performance. The RDYC register should be programmed before the use of the Advanced Protocol begins.

Set using the STMC command.

Read using SCNS, RDBACK0.

Bits 18-17 of the Global register state.

59.7.2.3.8 Test Reset (TRESET)

Asserts a test reset to the STL.

0 – The nSYS_TRST and SYS_TMS signals presented to the STL is not influenced by this bit.

1 – The nSYS_TRST presented to the STL is asserted and the SYS_TMS signal is a logic 1.

Set using the STC1 command.

The effects of the TRESET Register may be modified by a private TAP.7 controller register. When this private register is unimplemented or has a value of zero, the TRESET register causes the maximum initialization of the STL with nSYS_TRST and the SYS_TMS logic 1 value. Otherwise, the value of the private register may modify the function of the TRESET Register bit.

When the TRESET Register is a logic 1, the STL scan path appears broken as the state of CLTAPC remains *Test-Logic-Reset* while the EPUTAPC state progression continues. When the operation of the EPUTAPC and CLTAPC controllers is coupled, as with a T1 TAP.7, the state of these two TAPCs may be resynchronized to the *Run-Test/Idle* state by:

- An STC1 Command that sets the TRESET Register value to a logic 0.
- A stay in the Run-Test/Idle state of two or more SYS_TCK periods immediately following the Update-DR state of this command

A failure to follow the guidelines is considered a programming error and will result in erroneous system operation. The EPUTAPC and CLTAPC states will not be synchronized as they progress through the state diagram.

59.7.2.3.9 Auxiliary Pin Function Control (APFC[1:0])

This register enables TDI and TDO alternate functions when 2-pin TAP.7 operation is selected.

0x – TDI and TDO have JTAG function.

1x – TDI and TDO have alternate function.

Set using STC2 command.

Read using SCNS, RDBACK0.

59.7.2.3.10 Conditional Group Member (CGM)

Used by STC1, STC2, STTESTM, and EXC3 – EXC0 commands as the “condition” for updating registers.

Set and cleared using MCM and SCNB commands.

Read using SCNS, RDBACK0.

59.7.2.3.11 Scan Group Candidate (SGC)

This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used.

- **0** – Decoupling action.
- **1** – Coupling action.

Set and cleared using STMC, MSC, and SCNB commands.

Read using SCNS, RDBACK0.

59.7.2.3.12 Sampling Edge (SEDGE)

The Sampling Edge register defines the TCKC edge used to sample the TMSC input.

- **0** – Sample TMSC on the TCKC falling edge.
- **1** – Sample TMSC on the TCKC rising edge.

Set and cleared using STC1 command.

Read using SCNS, RDBACK0

59.7.2.3.13 Read Back 0 (RDBACK0)

Non-transport related register information.

Read using SCNS command.

Unimplemented registers shall read back as "0".

Table 59-10. RDBACK register format

Bit	Width	Register Mnemonic	Name
4:0	5	SCNFMT	Scan Format
6:5	2	PWRMODE	Power-control Modes
7	1	FRESET	Functional Reset
8	1	TRESET	Test Reset
9	1	SGC	Scan Group Candidate
		CGM	Conditional Group Member
11	1	SSDE	Scan Selection Directive Enable
13:12	2	TOPOL	Topology
14	1	SREDGE	Sampling Edge
16:15	2	DLYC	Delay Control
18:17	2	RDYC	Ready Control
20:19	2	APFC	Auxiliary Pin Functional Control
22:21	2	STCKDC	SYS_TCK Duty Cycle
31:23	11	Reserved	Read as a logic 0

59.7.2.3.14 Configuration Register 0 (CNFG0)

Table 59-11. Configuration Register 0 Format

Field	Bit #	Width	Name	Description
CNFG (Class)	3:0	4	CLASS[3:0]	TAP.7 Class
				0000 – TAP.1 or TAP.7 Class T0
				0001 – Class T1
				0010 – Class T2
				0011 – Class T3
				0100 – Class T4

Table continues on the next page...

Table 59-11. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
				0101 – Class T5
CNFG (Revision)	7:4	4	REV[3:0]	TAP.7 specification revision.
				0000 – Reserved
				0001 – IEEE Std 1149.7-2008
				0010 – 1111 - Reserved
CNFG (Format)	11:8	4	CNFGFMT[3:0]	Configuration Register Format
				xxx1 – CNFG0[31:12] are implemented
				xx1x – CNFG1 implemented
				x1xx – CNFG2 implemented
				1xxx – CNFG3 implemented
T1 Options	12	1	RDBKS	RDBACK supported
				0 – Unsupported
				1 – Supported
	13	1	TRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	14	1	FRESETS	TRESET supported
				0 – Unsupported
				1 – Supported
	17:15	3	PWRMODES[2:0]	PWRMODES[x] – Power Control Mode
				0 – Unsupported
				1 – Supported
	18	1	PCMR	PCMR – PM default POWRMODE value
				0 – No
				1 – Yes

Table continues on the next page...

Table 59-11. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
T2 Options	19	1	DCAS	DCAS – Decouple at start-up
				0 – No
				1 - Yes
T4 Options	21:20	2	SSCANS[1:0]	x0 – SScan1:0 Unsupported
				x1 – SScan1:0 Supported
				0x – SScan3:2 Unsupported
				1x – SScan3:2 Supported
	24:22	3	OSCANS[2:0]	xx0 – OScan3:2 Unsupported
				xx1 – OScan3:2 Supported
				x0x – OScan5:4 Unsupported
				x1x – OScan5:4 Supported
				0xx – OScan7:6 Unsupported
				1xx – OScan7:6 Supported
	25	1	APFCS	Auxiliary Pin Function Control supported
				0 – Unsupported
				1 – Supported
	26	1	TAPWIDS	TAP Width Default
				0 – 2-pin supported
				1 – 4-pin supported
	27	1	TAPWLCK	TAP Width Locked
				0 – Not locked
				1 - Locked
	28	1	STCKDCS	SYS_TCK Duty Cycle supported
				0 – Unsupported
				1 – Supported

Table continues on the next page...

Table 59-11. Configuration Register 0 Format (continued)

Field	Bit #	Width	Name	Description
Reserved	31:29	3	Reserved	Reserved, read as zero

59.7.2.3.15 Register reset values

Table 59-12. Reset values of TAP.7 controller registers managed with commands

Register type	Width	Register mnemonic	Name	Values for reset type			
				0	1 – 3	4	5
Control	1	ECL	Exit Control Level	0	0	0	NC
	1	SUSPEND	Suspend	0	0	0	
	1	ZBSINH	ZBS Detect Inhibit	0	0	0	
	5	SCNFMT	Scan Format	DF	DF	DF	
	1	SSDE	SSD Enable	0	0	0	
	2	DLYC	Delay control	00	00	00	
	2	RDYC	Ready Control	00	00	00	
Options	1	TRESET	Test Reset	0	0	0	NC
	2	APFC	Aux. Pin Functional Cntl.	DP	DP	NC	
Select	1	CGM	Conditional Group Member	0	0	0	NC
	1	SGC	Scan Group Candidate	DCA	DCA	DCA	
	1	SREDGE	Sampling Rising Edge	0	0	0	
Read-only	32	RDBACK0	Read-back 0	N/A	N/A	N/A	N/A
	32	CNFG0	Configuration Register 0				

NC = No Change

DF = Default Scan Format

DM = Default Power Control Mode

DCA = Default Coupling Action

DT = Default Topology

DP = Default Pin Function

59.7.3 EPU Commands

There are two command types:

- Two Part Commands
- Three Part Commands

Any number of two-part and three part commands may be issued in any combination before the command control level is exited. If the command control level is exited before a two- or three-part command is completed, the command is aborted and becomes a NOP.

59.7.3.1 Two Part Commands

TAP.7 controller commands are 10-bit values. The 10-bit commands are created by two consecutive DR-Scans when the control level is locked at the command control level (two). The first DR-Scan provides a five-bit command opcode. The second DR-Scan provides a five-bit operand. These two scans are called the command part one (CP1) and command part two (CP2), respectively. A command created entirely with these two scans is called a two-part command.

59.7.3.2 Three Part Commands

Some commands are used to send and/or receive data values other than values embedded in the command's operand. These commands are called three-part commands. With these commands, CP1 and CP2 are followed by an additional DR-Scan to transport a data value to or from an EPU scan path. When used for this purpose, the DR-Scan is called a Control Register Scan (CR-Scan). The CR-Scan is a minimum of zero bits in length with there being no maximum length. The CR-Scan path is described in SECTION.

59.7.3.3 Command Sequence

The two 5-bit values concatenated to create the 10-bit command represent the number of Shift-DR states between the *Capture-DR* and *Update-DR* states of CP1 and CP2. The count within CP1 creates the MSBs of the command while the count within CP2 creates the LSBs of the command. The command is decoded when the *Update-DR* state of the second DR-Scan is reached. A determination as to whether the command is a two- or three-part command is made at this point. The command format is shown in the following figure.

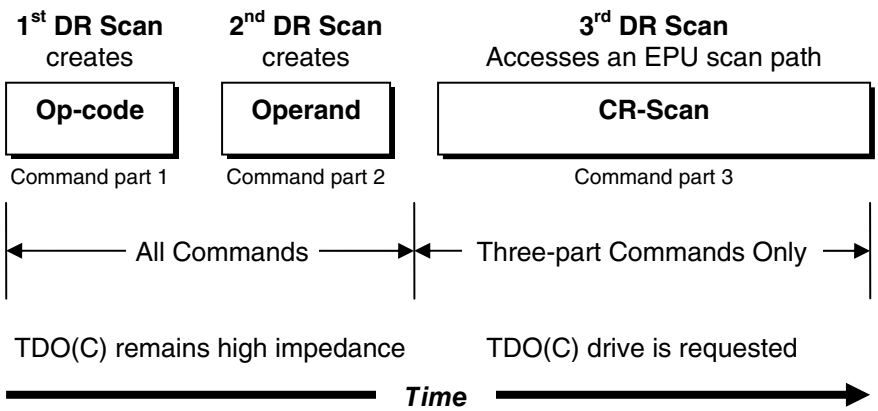


Figure 59-8. DR_Scan sequence used for command creation

The 10-bit command created by CP1 and CP2 determines the command's function and whether the command is a two- or three-part command. If the command is a two part command, the function specified by the command is performed when command part 2 completes. If the command is a three-part command, the DR-Scan following CP2 performs a TAP.7 controller function designated by the 10-bit controller command. This DR-Scan is called a ***Control-Register Scan*** (CR-Scan) and can be any length. A CR-Scan has many of the attributes of the DR-Scan as it moves data between the external tool and TAP.7 controller and accesses various EPU scan paths. There are many two part commands, but only three three-part commands, each having a special purpose.

59.7.3.4 Commands

There are 11 mandatory and 5 optional commands that can be grouped in classes as shown in the following list:

- Store – The operand is stored in a register or causes an action.
- Select – Controls participation of Scan and conditional command execution.
- Scan – Inputs and outputs with a CR-Scan.
- Enumerate – Controller ID allocation and de-allocation.
- Private – Commands available for chip-specific definition.

The definition of the commands with their operands is shown in the following table.

Table 59-13. – TAP.7 controller command list

Command class	Opcode count	Mnemonic	Name	TAP.7 class			
				1	2	3	4
Store	0x00	STMC	Store Miscellaneous Control	M	M	M	M
	0x01	STC1	Store Conditional 1-bit	M	M	M	M
	0x02	STC2	Store Conditional 2-bit	M	M	M	M
	0x03	STFMT	Store Format	-	M	M	M
	0x04	STDCST	Store Data Channel State	-	-	-	-
Select	0x06	MCM	Make Cond. Group Member	M	M	M	M
	0x07	MSC	Make Scan Group Candidate	-	M	M	M
Scan	0x08	SCNB	Scan Bit	M	M	M	M
	0x09	SCNS	Scan String	M	M	M	M
Enumerate[1]	0x0A	CIDA	Allocate Controller ID	-	-	M	M
	0x0B	CIDD	De-allocate Controller ID	-	-	M	M
Reserved	0x0C–0x1F	Reserved	Reserved	-	-	-	-

1. Enumerate commands are not currently supported by the CJTAG module.

59.7.3.4.1 Store Commands

Store commands are two-part commands that are available in both Star and Series Scan Topologies. The Store Miscellaneous Control (STMC) Command performs control functions that affect all TAP.7 controllers sharing an external tool connection. It stores values in one-bit and two-bit registers.

The Store Conditional 1 bit (STC1) and Store Conditional 2 bit (STC2) Commands store one-bit and two-bit values in registers, respectively. These commands operate two ways:

1. Unconditionally storing registers in all TAP.7 controllers or
2. Storing registers in only the TAP.7 controllers that have a logic 1 Conditional Group Member (CGM) Register value.

The Store Format (STFMT) Command unconditionally stores a five-bit Scan Format value.

The Store Data Channel State (STDCST) Command stores a five-bit value identifying the states supporting transport.

59.7.3.4.2 Select Commands

Select commands are two-part commands that are available in both Star and Series Scan Topologies. Their primary use is to select TAP.7 controllers of interest in a Star Scan Topology. These commands define the TAP.7 controller(s) participating in scans and the TAP.7 controllers that conditionally execute STC1 and STC2 Commands.

The Make Conditional Group Member (MCM) Command specifies which TAP.7 controllers execute store conditional commands as it manages the Command and Path Enable (CAPE) Register. This register also affects the execution of private commands and is utilized in EPU scan path selection.

The Make Scan Group Candidate (MSC) command manages the SGC register. This register determines whether the STL is coupled when the *Run-Test/Idle* state is reached when scan formats other than JScan0 and JScan1 are used. Select commands execute only when the CIDI Register is a logic 0, indicating the TAP.7 controller's CID is valid, and are treated as no operations otherwise.

59.7.3.4.3 Scan Commands

Scan commands are three-part commands that are available in both Star and Series Scan Topologies. The CR-Scan portion of Scan Bit (SCNB) and Scan String (SCNS) Commands is used to set and test bits (SCNB) or transfer strings of bits (SCNS). The commands may be used to access both public and private registers.

59.7.3.5 Command Definitions

59.7.3.5.1 Store Commands

59.7.3.5.1.1 Store Miscellaneous Control Command (STMC)

Table 59-14. Store Miscellaneous Control

STMC		Store Miscellaneous Control				
Opcode	**Operand**	**bbb**		**x**	**y**	**Description**
00000	bbb x y	000	StateCtl	0	0	NOP
				0	1	ExitCmdLev(ECL)
				1	0	Exit/Suspend (SUSPEND) = 1
				1	1	ZBS Inhibit (ZBSINH) = 1
		001	ScanCtl	0	-	SGC = y
				1	-	CGM = y

Table continues on the next page...

Table 59-14. Store Miscellaneous Control (continued)

STMC		Store Miscellaneous Control				
Opcode	**Operand**	**bbb**		**x**	**y**	**Description**
		010	RdyCtl	-	-	RDYC = xy
		011	DlyCtl	-	-	DLYC = xy
		100	ScnFunc	0	-	SSDE = y
				1	0	Star-4 Topology Test
				1	1	Reserved
		101	Reserved	-	-	Reserved
		110				
		111				

59.7.3.5.1.2 Store Conditional 1-bit (STC1)

Table 59-15. Store Conditional 1 bit (STC1)

STC1		Store Conditional 1 bit			
Opcode	**Operand**	**c**	**bbb**	**CAPE**	**Description**
00001	c bbb v	0	000	-	SREDGE = v
			001	-	Reserved
			010	-	TRESET = v
			011 – 111	-	Reserved
		1	–-	0	No change
			000	1	SREDGE = v
			001	1	Reserved
			010	1	TRESET = v
			011 – 111	1	Reserved

59.7.3.5.1.3 Store Conditional 2-bit (STC2)

Table 59-16. Store Conditional 2 bit (STC2)

STC2		Store Conditional 2 bit			
Opcode	**Operand**	**c**	**bb**	**CAPE**	**Description**
00010	c bb vv	0	00	-	Reserved
			01	-	STCKDC = vv
			10	-	APFC = vv
			11	-	Reserved
		1	–-	0	No change
			00	1	Reserved
			01	1	STCKDC = vv

Table continues on the next page...

Table 59-16. Store Conditional 2 bit (STC2) (continued)

STC2		Store Conditional 2 bit			
Opcode	Operand	c	bb	CAPE	Description
			10	1	APFC = vv
			11	1	Reserved

59.7.3.5.1.4 Store Scan Format (STFMT)

Table 59-17. Store Scan Format (STFMT)

STFMT		Store Scan Format
Opcode	Operand	
00011	nnnnn	SCNFMT = nnnnn

59.7.3.5.2 Select Commands

This section describes the select commands with their operands.

59.7.3.5.2.1 Make Conditional Group Member (MCM)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the CGM bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The CGM Register is also stored with the STMC and SCNB commands.

Table 59-18. Make Conditional Group Member (MCM)

MCM		Conditional Group Member	
Opcode	Operand	m	Description
00110	m iiii	0	CGM bit of the non-targeted controller is cleared
		1	CGM bit of the targeted controller is set. CGM bit of a non-targeted controller is unaffected.

59.7.3.5.2.2 Make Scan Group Candidate (MSC)

This command operates as a no operation when the CIDI Register is a logic 1 (The Controllers CID is invalid). Otherwise it sets the SGC bit in a TAP.7 controller that has a valid CID and the CID matches the operand CID field (iiii). The SGC Register is also stored with the STMC and SCNB commands.

Table 59-19. Make Scan Group Candidate (MSC)

MSC		Make Scan Group Candidate	
Opcode	**Operand**	**m**	**Description**
00111	m iiii	0	SGC bit of the non-targeted controller is cleared
		1	SGC bit of the targeted controller is set. SGC bit of a non-targeted controller is unaffected.

59.7.3.5.3 Scan Commands

59.7.3.5.3.1 Scan Bit (SCNB)

This is a 3-part command. The operand chooses the bit to be read or written by the CR-Scan.

Table 59-20. Scan Bit (SCNB)

SCNB			Scan Bit		
Opcode	**Operand**	**CR-Scan**	**yyyyy**	**Description**	**R/W**
01000	yyyyy	-	0	Scan Group Candidate (SGC)	Write only
		-	1	Conditional Group Member (CGM)	Write only
		-	2	CNFG[0]	Read only
		-	3	CNFG[1] (not implemented)	Read only
		-	4	CNFG[2] (not implemented)	Read only
		-	5	CNFG[3] (not implemented)	Read only
		-	6	Functional reset requested (FRESETR) (not implemented)	Read only
		-	7	Series topology detect	Write only
		-	15 – 8	Reserved	Read only

59.7.3.5.3.2 Scan String (SCNS)

This is a 3-part command. The operand chooses the string to be read or written by the CR-Scan.

Table 59-21. Scan String (SCNS)

SCNS			Scan String		
Opcode	**Operand**	**CR-Scan**	**yyyyy**	**Description**	**R/W**
01001	yyyyy	-	0	RDBACK[0] Register [31:0]	Read only

Table continues on the next page...

Table 59-21. Scan String (SCNS) (continued)

SCNS			Scan String		
Opcode	Operand	CR-Scan	yyyyy	Description	R/W
		-	1	RDBACK[1] Register [31:0] (not implemented)	Read only
		-	2	CNFG[0] Register [31:0]	Read only
		-	3	CNFG[1] Register [31:0] (not implemented)	Read only
		-	4	CNFG[2] Register [31:0] (not implemented)	Read only
		-	5	CNFG[3] Register [31:0] (not implemented)	Read only
		-	6 - 7	Reserved (read logic 0)	Read only
		-	31 - 8	Private – available for customization	

59.7.4 EPU operating states

The TAP.7 controller and STL share the use of ZBSs. The EPU has three operating states as shown in the following figure:

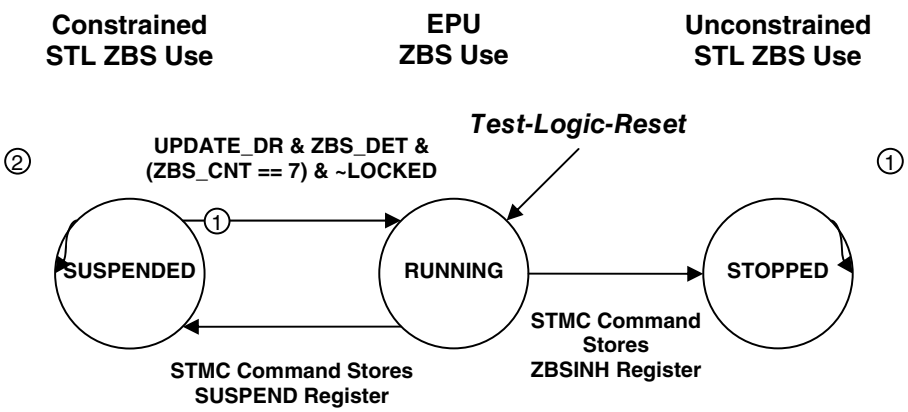


Figure 59-9. EPU operating states

The values of the SUSPEND and ZBSINH Registers determine the type of operation as shown in the following table. The Test-Logic-Reset state sets the SUSPEND and ZBSINH Register values to a logic 0.

Table 59-22. ZBS use state characteristics with both TAP.7 and other ZBS use

ZBSINH	SUSPEND	EPU operating State	ZBS use by:	Until
0	0	RUNNING	EPU	Command allocates to STL

Table continues on the next page...

Table 59-22. ZBS use state characteristics with both TAP.7 and other ZBS use (continued)

ZBSINH	SUSPEND	EPU operating State	ZBS use by:	Until
0	1	SUSPENDED	STL	ZBS sequence allocates to EPU
1	0	STOPPED	STL	Test-Logic-Reset state allocates to EPU
1	1	Not possible		

59.7.5 System and EPU Paths

The System path is utilized when the external tool requires access to system (STL) resources. This path is used when either the SUSPEND Register is a logic 1 or the ZBS count is less than or equal to one. The EPU path is used otherwise.

The EPU scan path is constructed from five types of EPU scan paths:

- **Bypass** – A 1-bit scan path
- **Bit** – Supports set and test bit operations (a 1-bit shift register)
- **String** – Support 32-bit reads or 32-bit writes
- **Enumerate** – A 36-bit scan path that transports a TAPC address with T3 and above TAP.7s
- **Auxiliary** – An n-bit scan path that transports 1 – n bits using a conventional IEEE 1149.1 style DR-Scan path using Control Levels 4 and 5.

59.8 APU (Advanced Protocol Unit) Operation

59.8.1 Overview

The Advanced Protocol Unit (APU) is placed between the EPU and the RSU to create the T4 TAP.7 controller. The APU delivers advanced capabilities with a number of operating modes (new scan formats) that utilize the Standard Protocol and the Advanced Protocol in both the narrow and wide T4 TAP.7 versions.

The APU enables scan transfers with two signals by serializing the information related to a TAPC state machine state. It provides a number of serialization formats to balance scan flexibility and scan performance. It multiplexes the use of the TMSC signal for T4 TAP.7 controller functions. The serialization is bypassed to provide T3 TAP.7 capability.

With a wide TAP.7, the chip architect may choose to implement the TDIC and TDOC signal functions as fixed functionality (the T3 TDIC and TDOC functions) or as programmable between the T3 TDIC and TDOC signal functions and an alternate function using a TAP.7 controller register. In this case, the default signal function may be either of the programmable choices. Programmable TDIC and TDOC functionality provides a means to fully utilize these signals in any scan topology as the T3 TAP.7 functionality is fully supported.

The entire EPU infrastructure (i.e., ZBS detection, control-level management, and command generation) is utilized when either the Standard Protocol or the Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state machine state progression. The EPU is unaware of the APU's existence. All T4 features are managed outside of the EPU.

The TAP.7 serialization schemes are specified with scan formats that complement the JScan 0-3 Scan Formats inherited from the T3 TAP.7. The scan formats are called advanced scan formats as they support two-signal operation.

Two groups of scan formats (MScan and OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. The OScan Scan Format is broken into two groups with the same function, one group has better performance but may only be used with an external tool sourced TCKC. The external tool chooses a scan format that matches the constraints imposed by the application, chip components, and possibly the external tool itself.

The MScan and OScan formats address Test and Debug use cases. The mandatory MScan and OScan0-1 formats emphasize flexibility over performance. The remaining optional Oscan2-7 formats emphasize performance over flexibility by not transmitting unneeded information such as TDI and TDO in non-shift TAPC state machine states.

The optional capabilities added with T4 and above TAP.7s change the bit sequences seen at the TAP.7 signals. A TAP.7 controller comprehends only the bit sequences generated by the use of the features it supports. It is never exposed to bit sequences it does not comprehend. A check for supporting an enabled feature is made before the feature is used (alters the bit sequences). When a T4 TAP.7 controller detects the use of an unsupported scan format, it halts its operation and places itself in a state where its operation may be resumed. This state is henceforth called "offline". A TAP.7 controller that is not offline is henceforth called "online". While offline the TAP.7 controller awaits the detection of a

special signaling sequence called the Online Activation Protocol. This protocol sequence places an offline TAP.7 controller online and synchronizes the operation of all online TAP.7s.

The Advanced Protocol Unit supports the stall of a transfer by both the external tool and STL at any TAPC state.

The APU provides the following functions:

- Bypass (for operation with the Standard Protocol)
- Scan control (for SP use)
- Configuration Control (for CP use)

The Scan Control, Configuration Control, and Transport Control functions are idled when the Standard Protocol is used as the Bypass Function connects the TCKC and TMSC pins to the EPU.

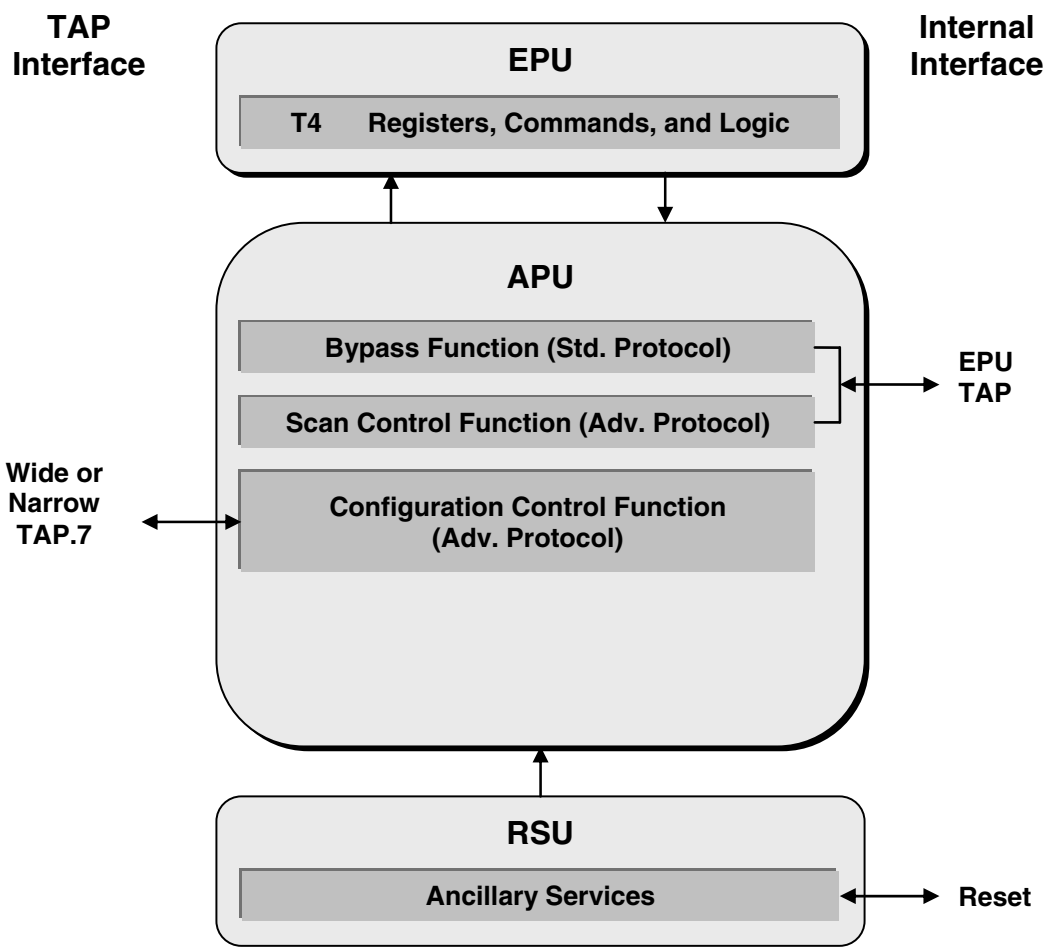


Figure 59-10. Conceptual block diagram of the APU functions

The APU's internal interface includes a connection to EPU registers, the EPU's TAP interface, connections to chip-level data channels (when they are implemented), and reset signaling with the EPU. The APU's TAP interface provides mutually exclusive use of the TMSC pin for:

- Bypass Function
- Scan Control Function
- Configuration Control Function

When the Standard Protocol is used, the APU's Bypass function connects the EPU directly to the TAP.7 pins. When the TDIC and TDOC pin functions are not available (narrow TAP.7 or TDIC and TDOC pin functions are an alternate function), the APU sources a logic 1 to the EPU inputs associated with these pins.

When the Advanced Protocol is used, the APU multiplexes the use of the TMSC pin between the other APU functions. Scan Control acts as a serial to parallel converter, translating the parallel information at the EPU's TAP interface into bidirectional serial transfers. Since the Scan Control is literally a protocol adaptor, the entire EPU infrastructure (ZBS detection, control-level management, and command generation) is utilized when either the Standard or Advanced Protocol is used, as these TAP.7 controller attributes are controlled entirely from the TAPC state progression.

59.8.2 Operation

The conceptual view of the APU operation is shown in the following figure.

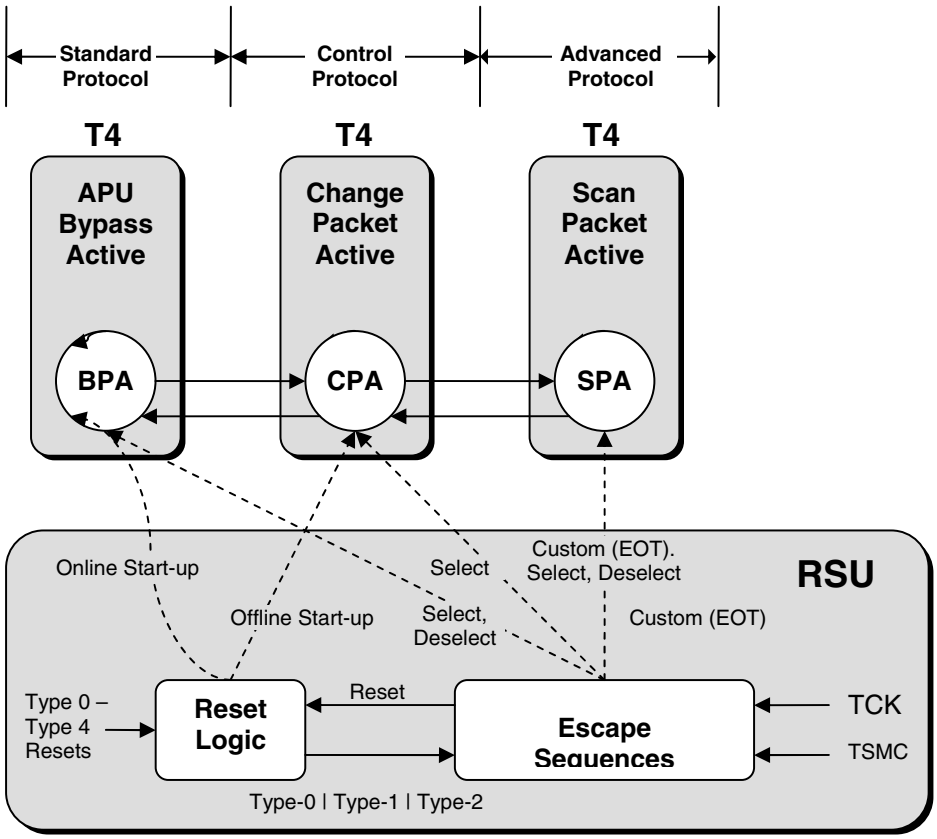


Figure 59-11. Conceptual view of APU operation

The APU operating states managing the use of the TAP pins and the ancillary (RSU) services use of the TMSC pin.

APU operating states:

- ***BPA*** – **B**y**P**ass **A**ctive
- ***CPA*** – **C**hange **P**acket **A**ctive
- ***SPA*** – **S**can **P**acket **A**ctive

59.8.2.1 BPA

The *BPA* state allocates the use of the TMSC pin to the Standard Protocol. This state is entered two ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up online.
- From the CPA state when a TAP.7 controller register write specifies the use of the Standard Protocol.

59.8.2.2 CPA

The *CPA* state allocates the use of the TMSC pin to a CP. This state is entered in one of three ways:

- As a result of a TAP.7 controller reset when the TAP.7 controller starts up offline.
- A value specifying the use of the Advanced Protocol is written to the SCNFMT Register when using the Standard Protocol.
- A TAP.7 controller register write occurs when using the Advanced Protocol.

59.8.2.3 SPA

The SPA state allocates the use of the TMSC pin to an SP. This state is entered when the SCNFMT Register specifies the use of the Advance Protocol when a CP ends.

59.8.3 Escape sequences

Escape-sequence detection provides services to the Reset Logic, Scan, Configuration Control, and Transport Control functions as outlined below:

- **Scan and Transport Control** – End of Transfer (EOT) escape sequence detected.
- **Configuration Control** – Synchronize Advanced Protocol (SAP) escape sequence.
- **Reset Logic** – Reset (RES) escape sequence detected.

An EOT escape sequence is used to increase the efficiency of OScan Scan formats.

An SAP escape sequence is used to place TAP.7 controllers online.

A RES escape sequence asynchronously causes generation of a Type-3 Reset. Following a RES escape sequence, the drive of the TMSC pin is inhibited so it is not driven when TCKC becomes a logic 0 immediately following the RES escape sequence.

59.8.4 Signal behaviors

The TCKC signal:

- The test clock

The TMSC signal:

- An input when the Standard Protocol is used
- Bi-directional when the Advanced Protocol is used

The TDI and TDO signals:

- Deleted with a narrow TAP.7
- One of two functions with a wide TAP.7
  - Fixed – as the TAP.7 TDIC and TDOC pin functions, or
  - Programmable – as the TAP.7 TDIC and TDOC signal functions or alternate functions with a TAP.7 controller reset establishing the default signal function.

59.8.5 APU Functions

59.8.5.1 Bypass Active (BPA) Function

When the Standard Protocol is used, the APU connects the EPU inputs and outputs directly to the TAP.7 pins. If the TDIC and TDOC pin functions are not present (e.g., a narrow TAP.7 or the TDIC and TDOC pins have an alternate function assigned to them), the APU provides a logic 1 to the EPU for unsupported pin functions and ignores the EPU's TDO output data.

59.8.5.2 Scan Packet Active (SPA) Function

The Scan Control function is encapsulated within the Scan Packet Active (SPA) state. With the Standard Protocol, the external tool and TAP.7 exchange the scan information associated with a single TAPC state machine state in one TCK period using the TMS(C), TDI(C), and TDO(C) pins. With the Advanced Protocol all or part of this information is exchanged serially within a Scan Packet (SP) as shown in . There is a one to one correspondence between SPs and TAPC state machine state changes, except when the SPA state precedes the CPA state. In this case the SP preceding a CP does not advance the TAPC state machine state.

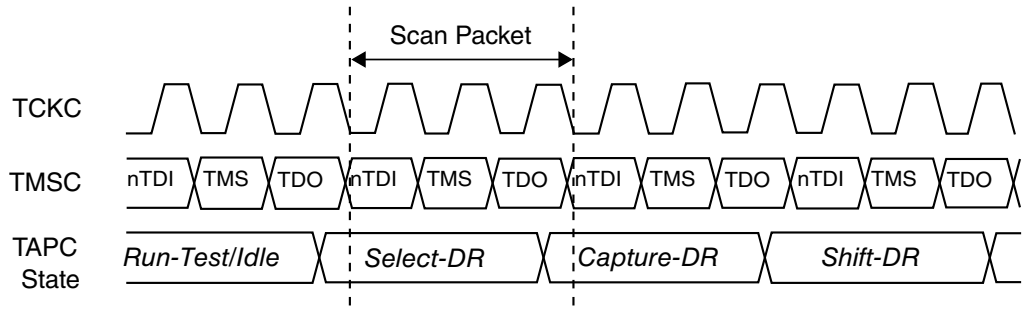


Figure 59-12. Scan Packet Sequence

A comparison of the parallel transfer of TAP information with the Standard Protocol and the serialization and de-serialization of the TAP information using the TCKC and TMSC pins with the Advanced Protocol is shown in FIGURE. The IEEE 1149.1 signals between the EPU and STL are prefixed with SYS_.

The APU manages the serialization and de-serialization of the scan information. The APU converts the TDI and TMS information within a Scan Packet (SP) into the TMS and TDI information presented to the EPU. It converts the combination of the EPU TDO and EPU TDO_OE signals into the TDO information in the same SP. If the EPU indicates that the TDO pin drive is not enabled, logic 1 TDO data is created, otherwise the TDO data is the supplied by either the EPU or STL.

Information within a Scan Packet is first passed from the external tool to the TAP.7 followed by information passed from the TAP.7 to the external tool. In this example the scan format specifies the exchange of the TMS, TDI, and TDO TAPC information. Each TAPC state machine state takes three TCKC periods. Other scan formats specify the exchange of different mixes of information. Control information is added with the most flexible scan formats. Some scan formats send less information and consequently take fewer TCKC periods.

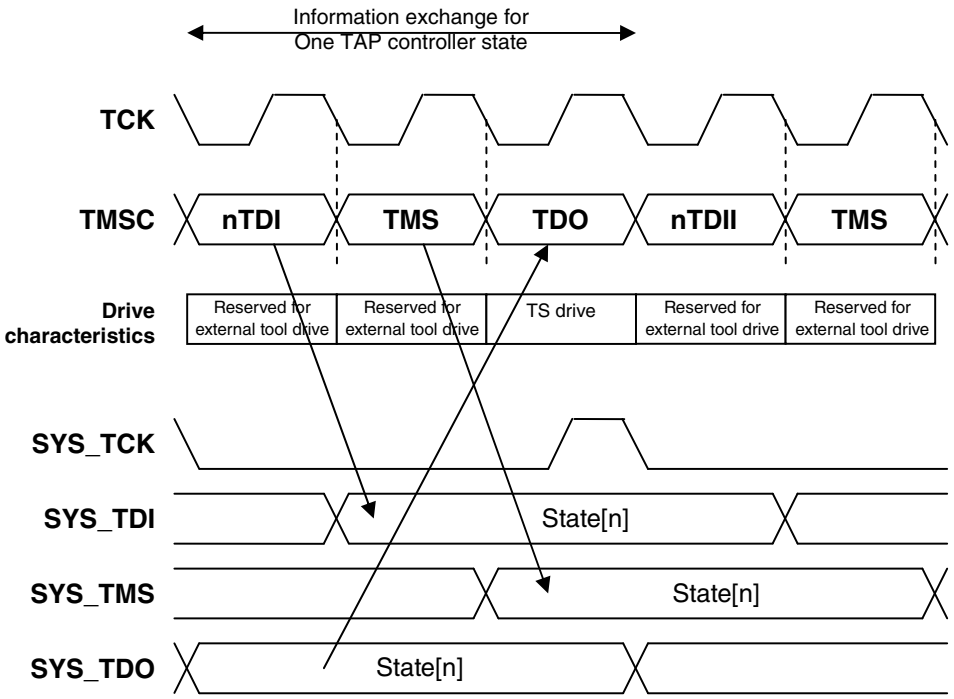


Figure 59-13. Serialization and De-serialization of Scan Packet Information

In order to produce comparable TAPC state machine state rates with a TAP.1 and TAP.7 controller:

- Full-cycle timing between the external tool and TAP.7 allows operation at higher TCKC frequencies.
- The number of bits transferred to advance a TAPC state machine is minimized. (reaching as few as one for some scan formats.)

The TAP.7 controller provides both full-cycle bit timing (falling TCKC edge to falling TCKC edge) and half-cycle bit timing (falling TCKC edge to rising TCKC edge), with a TAP.7 register used to define the type of timing used. Half-cycle timing is used as the default following a TAP.7 controller reset. The TCKC frequency is reduced to accommodate half-cycle timing.

With full-cycle timing, the TMSC input and output data are both output with and sampled with the falling edge of the TCKC. This means that the entire TCKC period may be spent transferring a bit from the external tool to the TAP.7 or vice-versa. The TCKC frequency can therefore be maximized when full-cycle timing is used. With half-cycle timing, the APU samples the TMSC input data with the rising edge of the TCKC and outputs TMSC data with the falling edge of TCKC.

Minimizing the number of bits transferred per TAPC state machine state is affected by a number of factors. The information that must be transferred per TAPC state machine state varies for different test and debug applications. Also, with some test and debug

applications, all TAPC state machine states do not require that the same amount of information be transferred. This is especially true with debug applications where specialized use of the TAP is more likely.

59.8.5.3 Control Process Active (CPA) Function

While using the Advanced Protocol, all TAP.7 configuration changes occur within the CPA state. All online and offline transitions also occur while in the CPA state. When entering the CPA state from the BPA or SPA state, the CPA state checks for the use of supported features affecting the Advanced Protocol. The CPA state is exited when the enabled features affecting the Advanced Protocol are supported.

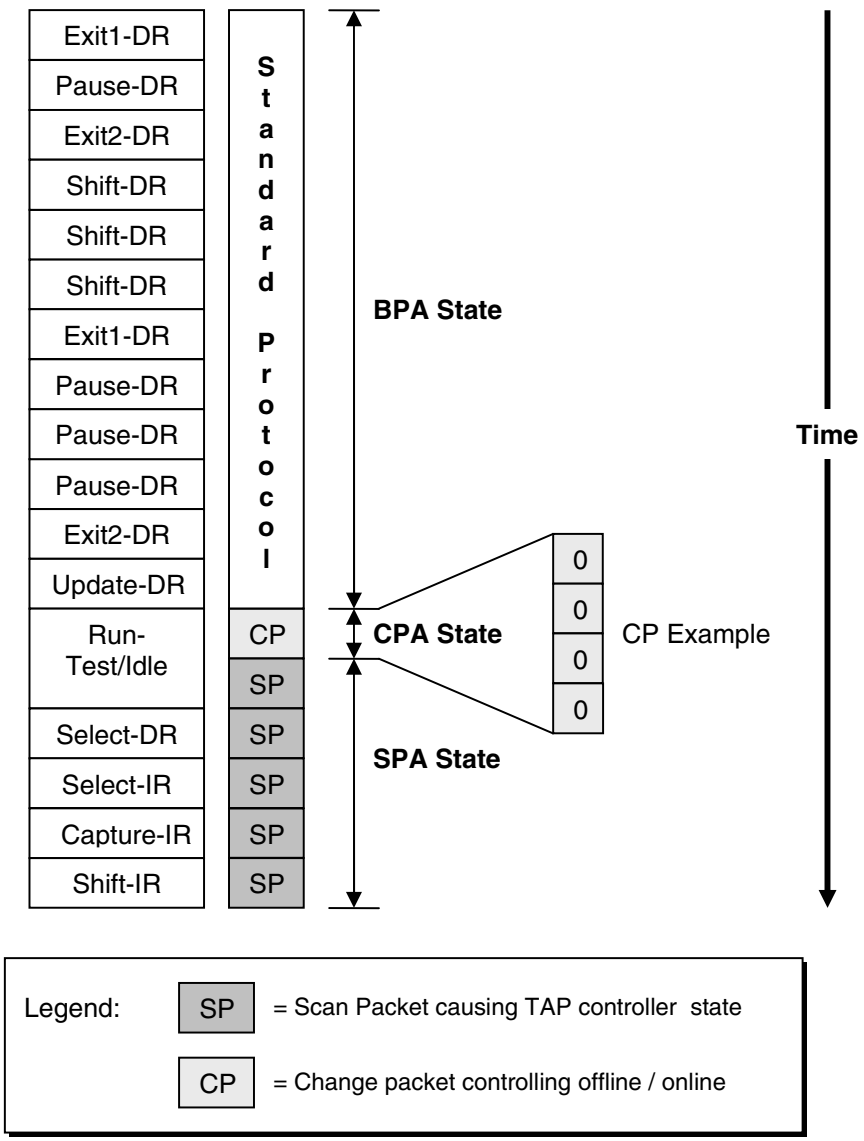


Figure 59-14. CPA State Entry from BPA State with Continued Online Operation

An SP preceding a CP does not advance the TAPC state machine state. This SP performs housekeeping functions that assure the contents of the SP preceding this SP have been fully utilized. This has significance when the SP contains control information that allows the STL to stall the completion of a SP.

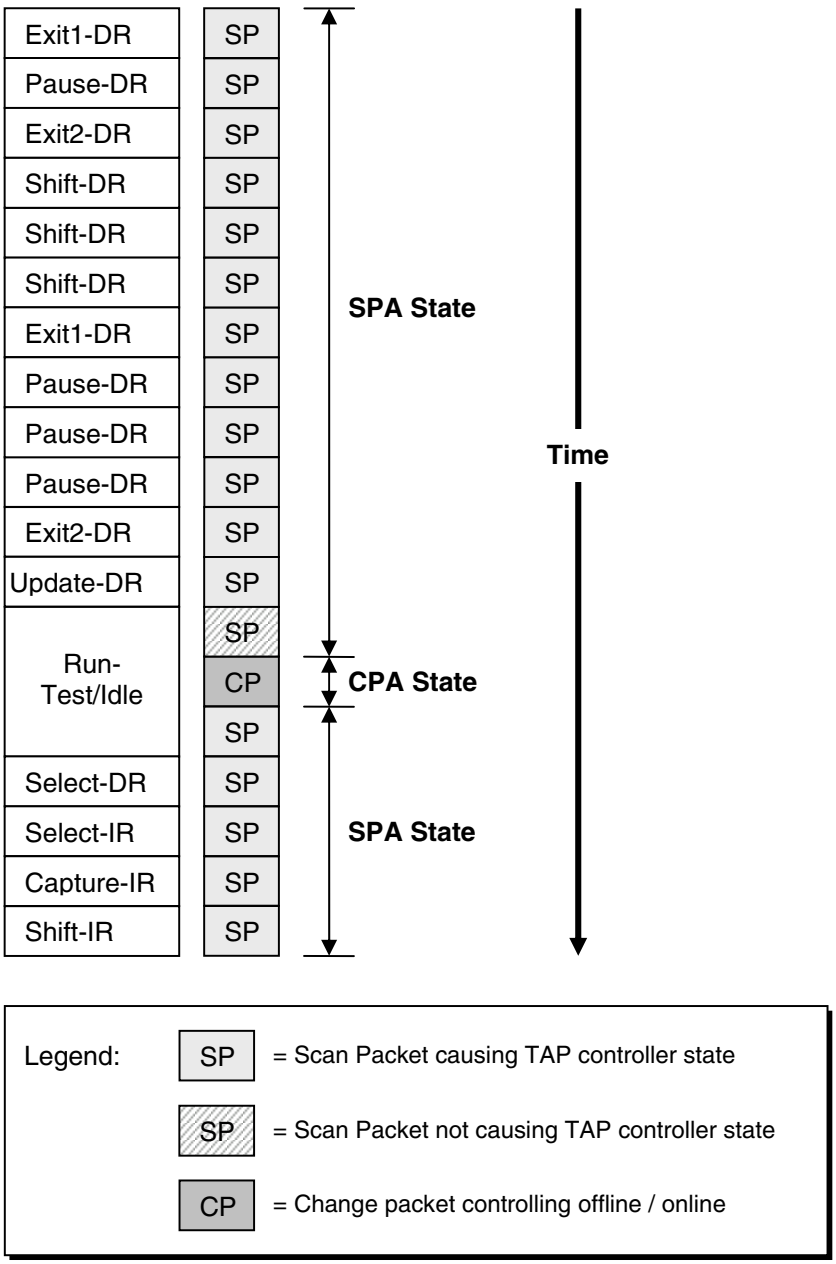


Figure 59-15. CPA State Entry from the SPA State with Continued Online Operation

When an unsupported feature is enabled, the TAP.7 controller is placed offline. This suspends the TAP.7 controller's operation at a point where its operation may later be synchronized to the remainder of the system. The APU stops advancing the EPU's and CLTAPC's TAPC state machine state while offline. It also suspends Data Channels activity while offline. The detection of the appropriate Online Activation Protocol while

the TAP.7 controller is offline places the TAP.7 controller online and causes an exit from the CPA state to the SPA state. An offline to online transition initiates the use of a predefined set of mandatory features in all online TAP.7 controllers to assure they operate synchronously.

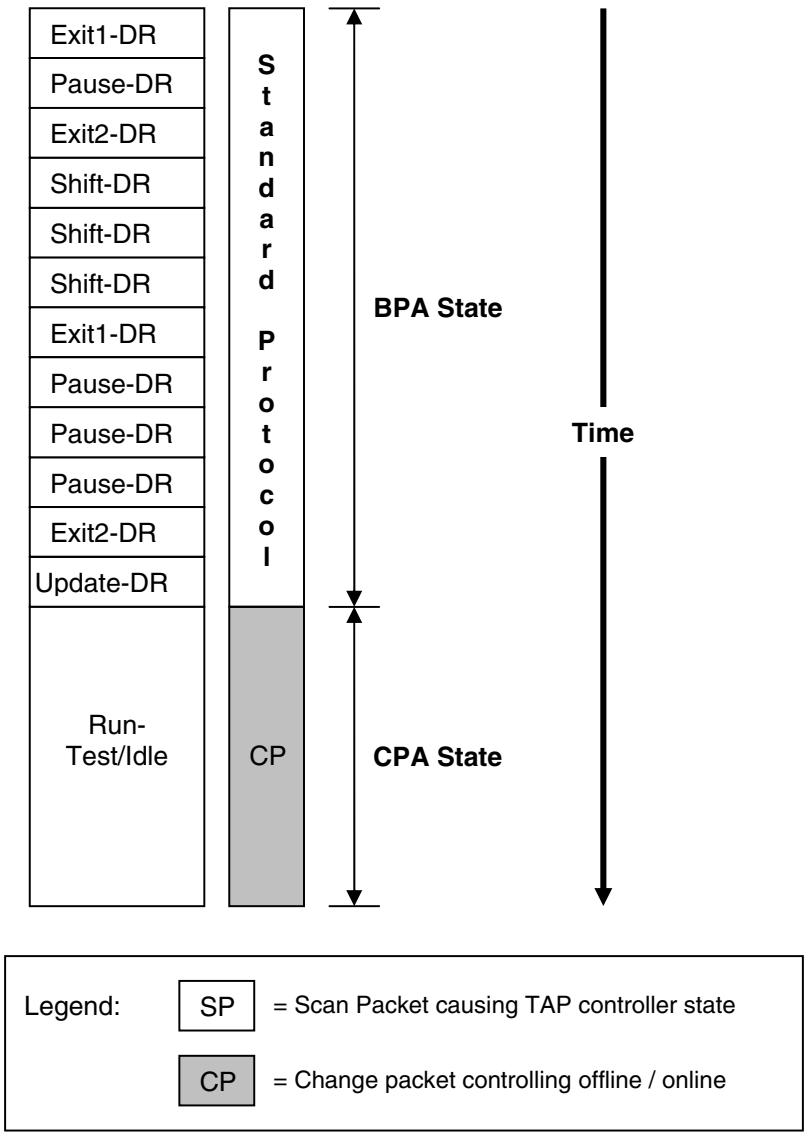


Figure 59-16. CPA State Entry from the BPA State with Offline Operation

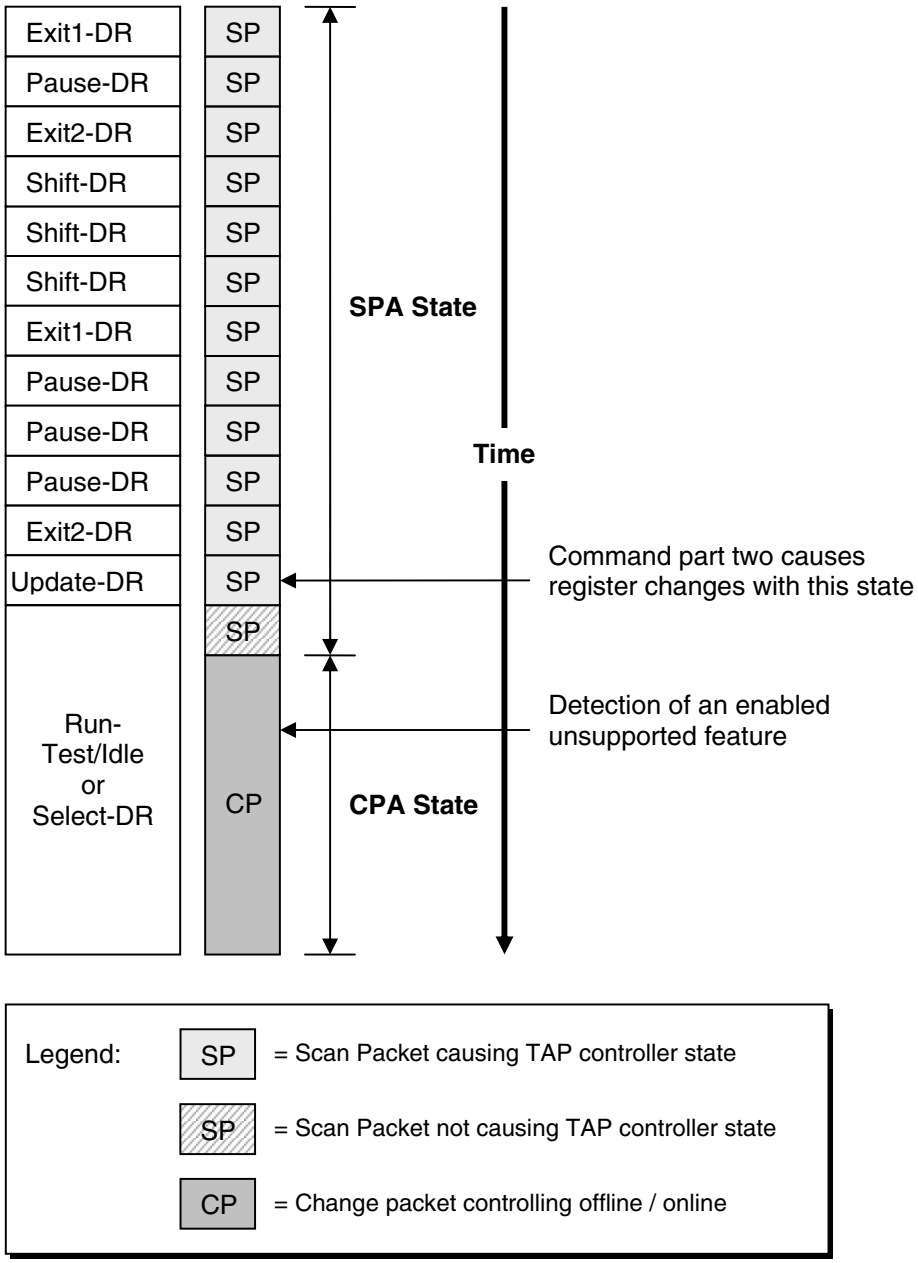


Figure 59-17. CPA State Entry from the SPA State with Offline Operation

A TAP.7 controller reset places the TAP.7 controller offline when the offline at start-up option is used. From this point the TAP.7 controller awaits being placed online.

59.8.6 Configuration Change Packets (CP)

A Change Packet (CP) is composed of three element types:

- **Preamble** – A 1-bit value that is a don't care. This bit facilitates a Standard to Advanced Protocol change with some external tool implementations.

- **Body** – Two or more bits representing one or more CP directives
  - **00** – CP_END
  - **01** – CP_NOP (extends the body by 1-bit)
  - **10** – CP_NOP (extends the body by 1-bit)
  - **11** – CP_RES (Generate a Type-3 TAP.7 controller reset coincident with the post amble bit.
- **Post amble** – A 1-bit value that is a don't care. This bit facilitates and Advanced to Standard Protocol change with some external tool implementations.

A minimum-length CP has a Preamble, a 2-bit Body, and a Post amble. This can be as simple as four logic zero values.

59.8.7 Scan Packet

A Scan Packet defines the period of TMSC pin activity where control information exchanged by the external tool and TAP.7 controller affects the TAPC state, except when the SP precedes a Change Packet. The external tool should view the Scan Packet as:

- Exchanging only the necessary TAPC information (TMS, TDI, and TDO or some subset thereof)
- Causing an advance of the TAP.7 TAPC state when it is not followed by a Change Packet.
- Causing an advance of the STL TAPC state when it is not followed by a Change Packet and when the STL is coupled.

With this perspective, the external tool views Scan Packets as running a virtual TAPC state machine at the TAP.7 pins. The effects of each Scan Packet take effect when the Scan Packet completes. The real TAP.7 and STL controller state machine states may change state before or after the virtual state machine.

59.8.8 SP Format

A SP serially exchanges the TDI, TMS, and TDO or a subset thereof, between the external tool and TAP.7 controller. Control information may be added to this exchange. A SP has three parts:

- **Payload** – Data + Control information (all scan formats), the payload content is varied to trade-off performance and flexibility. RDY bit(s) within the payload are included with some scan formats to provide the STL a means to stall progression of the TAPC state.
- **Delay** – Control information (optional) providing separation of adjacent payloads in time. Delay elements support simple external tool implementations where a delay element is used to stall progression of the TAPC state while it develops a new SP following the completion of the prior payload.

The header elements, delay elements, and control information within the payload element is consumed by the TAP.7 controller. This and other control information within the SP payload is not visible to the EPU and STL TAPCs.

59.8.8.1 Payload Element Content

59.8.8.1.1 Input and Output Bit Frames

An input bit frame is sourced by the external tool and an output bit frame is sourced by the TAP.7 controller. The payload may contain and input bit frame, an output bit frame, or both. When an input bet frame arrives at the TAP.7 controller, the TDI and TMS information within it is presented in parallel to the TAP.1 controllers within the EPU and STL. The output bit frame begins when the input frame is completed. The payload is completed when the EPU and STL are ready to utilize this information and the TDO data needed to complete the output bit frame is available. This process is repeated for each SP.

With input bit frames, MScan and OScan scan formats have no control bits input bit frames.

With output bit frames:

- TDO information passed to the external tool is a logic 1 when the EPU indicates the TDO pin would be high impedance.
- RDY bits provide TAP.7 stall opportunities with MScan and some OScan formats.
- PC0 and PC1 bits are added to MScan scan formats to allow Wire-ANDed output.

59.8.8.1.2 Varying the content of input and output bit frames

The various scan formats (MScan, OScan0-7) provide a number of options for the serialization of scan information for use in a Star-2 scan topology. These scan formats vary the content of SPs to balance flexibility and performance for different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. The MScan and OScan scan formats determine the factors affecting the payload content as shown in the following table.

Table 59-23. Factors determining the SP payload content

Scan format	Payload content determined by the:	
	TAPC state	Header
MScan	–	–
OScan	Yes	–

59.8.8.2 Delay Element Content

A delay element can be zero, one, two, or n bits in length. One and two bit delay elements are fixed length with neither the external tool or TAP.7 controller driving the TMSC pin. The TMSC pin is kept at the last value driven by either the external tool or TS. With an n-bit delay elements, the external tool drives the TMSC pin for the first n-1 bits and may drive the TMSC pin the nth bit. With an n-bit delay element the string of delay element bits is a series of directives virtually the same as those used by a CP (DLY_NOP, DLY_END, and DLY_RES).

59.8.8.3 Scan Formats

59.8.8.3.1 Overview

Scan formats utilizing the Advanced Protocol address different use cases. The amount of information transferred per TAPC state is based on the constraints imposed by these use cases. In one extreme two types of control information, the first allowing the stall of the completion of the Scan Packet, and the second supporting the Wire-AND of TAP.7 controller output are added to the TMS, TDI, and TDO information within the Scan Packet. In the other extreme, the external tool may minimize the information transferred for each TAPC state using its understanding of the application and transfer only the data of interest. In some cases the transfer of only one bit of information per TAPC is

required. There are a number of scan formats that operate between these two extremes. The external tool chooses a scan format that best matches the performance/functionality needs of the application.

The scan formats added to specify the use of the Advance Protocol are listed below:

- **MScan** – Maximum flexibility.
- **OScan0-7** – Optimized for test and debug applications.

These scan formats provide seven different functions. Five factors have influenced the definition of these scan formats:

- Supporting the capabilities of the IEEE1149.x and IEEE 1532 Standards.
- Providing the STL and external tool an opportunity to stall the TAPC state progression.
- Providing a minimum pin count and scalable functionality.
- Maximizing the efficiency of scan operations utilized for applications debug operations.
- Providing a rich set of capability for high end applications debug.

The performance and flexibility of MScan and OScan scan formats are shown in Table 59-24 and Table 59-26. The performance and flexibility of these scan formats are given subjective/approximate ratings in these tables. These ratings range from least to best. Use cases for these scan formats are also included in these tables.

59.8.8.3.2 MScan scan formats

The MScan scan format provides a universal scan function. It is the most flexible but lowest-performance scan format. The Scan Packets used by these scan formats are packed with data and control information and is the same for all TAPC states. This scan format accommodates the deployment of IEEE 1149.1 IP with non-compliant behavior within the STL. The STL is allowed to stall the TAPC state progression (to pace the arrival of Scan Packets with its ability to process them), and uses Wire-ANDed Output.

Table 59-24. MScan scan format use case summary

Scan format	Supporting	Performance	Flexibility	TCKC source
MScan	STL Stalls	Least	Best	External tool or TS
	Test			
	Non-compliant behavior STL IP			

Table continues on the next page...

Table 59-24. MScan scan format use case summary (continued)

Scan format	Supporting	Performance	Flexibility	TCKC source
	Multi-chip debug communication			
	Controller ID Allocation			

Table 59-25. MScan Scan Packet content

	Non-shift TAP controller states							Shift TAP controller states					
MScan	nTDI	TMS	PC0	RDY	PC1	TDO		nTDI	TMS	PC0	RDY	PC1	TDO

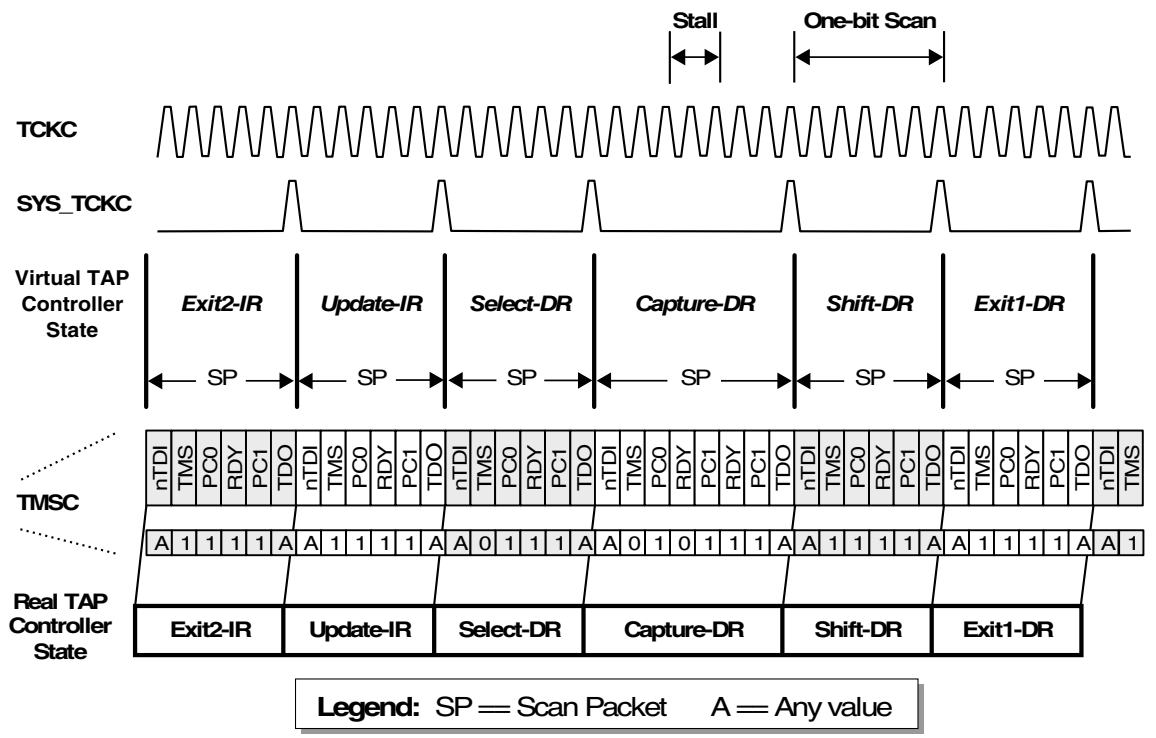


Figure 59-18. MScan Transaction

59.8.8.3.3 OScan Scan Formats

The OScan scan formats provide optimization choices where no knowledge of the data content of the scan exchange is needed. The Scan Packet content of these scan formats is modulated by a combination of the scan format and the TAPC state for some OScan scan formats. For instance, TDI data and TDO data is not exchanged for non-shift TAPC states and included in shift TAPC states in certain OScan formats. The modulation of Scan Packet content with TAPC state varies by scan format.

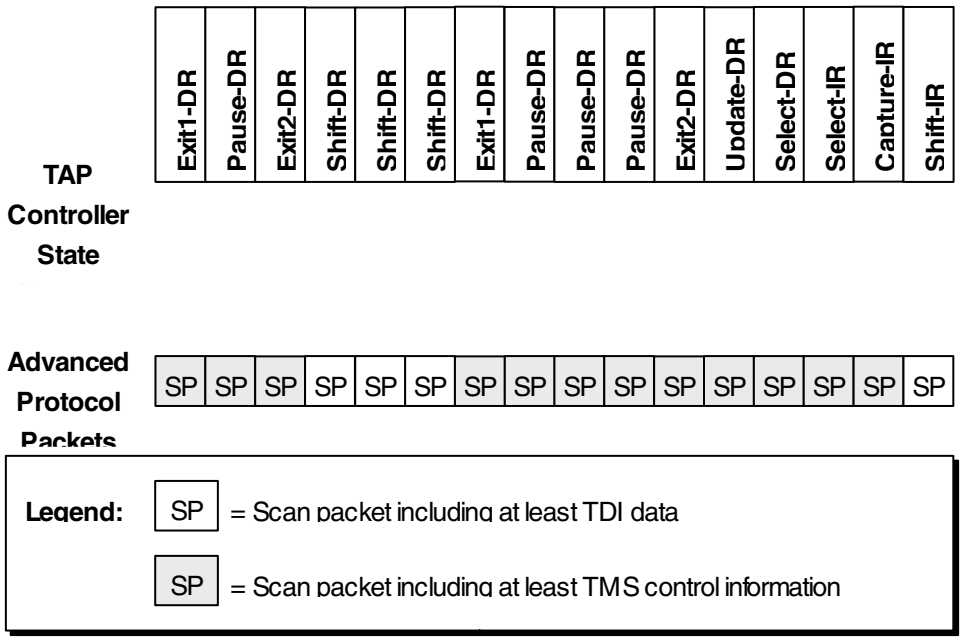


Figure 59-19. OScan Scan Packet content/TAPC state relationships

OScan scan formats replicate four functions in two forms:

- **OScan0 – OScan3:** Providing the best performance for a test clock source by the target system.
- **OScan 4 – OScan7:** Providing the best performance for a test clock sourced by the external tool.

The OScan4 – OScan7 scan formats are more efficient, as EOT escape sequences are used to identify an exit from the Shift-xR TAPC state. A Scan Packet associated with the Shift-xR TAPC state does not include a TMS bit. Instead, and EOT escape sequence overlaid on an nTDI bit causes an exit from shift TAPC state with no exit from this state occurring otherwise. A look at the 32-bit DR-Scan shows the impact of this optimization. A total of 96 bits are required when each Scan Packet associated with a Shift TAPC state when TMS is included in these Scan Packets while a total of 64 bits are required when TMS is not included in these Scan Packets. Assuming the EOT escape adds the equivalent of three bit periods, the number of TCK periods fo this transfer is 67/96 or roughly 70% of the total without this optimization.

These scan formats may only be used with an external tool sourced TCKC as EOTs are utilized to improve its performance. The OScan0 – OScan3 Scan Formats always have a TMS bit included in the Scan Packet. The OScan scan format use case summary is shown in the following table.

Table 59-26. OScan scan format use case summary

Scan Format	Supporting	Performance	Flexibility	TCKC Source
OScan0	Stalls	↓	Best	External tool
OScan1	Test		↑	
OScan2	Debug Performance			
OScan3	Debug Downloads (input only)	Best		
OScan4	Stalls	↓	Best	External tool or TS
OScan5	Test		↑	
OScan6	Debug Performance			
OScan7	Debug Downloads (input only)	Best		

Certain OScan optimizations are applied in a hierarchical manner.

- **Optimization I:** OScan1/OScan5 – RDY bit(s) and their trailing ONE bits are deleted from all Scan Packets.
- **Optimization II:** OScan2/OScan6 – Optimization I + nTDI and TDO bits are deleted from Scan Packets associated with non-shift TAPC states.
- **Optimization III:** OScan3/OScan7 – Optimization II + TDO bits are deleted from Scan Packets.
- **Optimization IV:** OScan4 – OScan7 – TMS bits are deleted from Scan Packets associated with shift TAPC states.

Table 59-27. OScan Scan Format "Payload"

Scan Format	Non-shift TAP controller states					Shift TAP controller states			
OScan0	nTDI	TMS	RDY	TDO		nTDI	TMS	RDY	TDO
OScan1	nTDI	TMS	–	TDO		nTDI	TMS	–	TDO
OScan2	–	TMS	–	–		nTDI	TMS	–	TDO
OScan3	–	TMS	–	–		nTDI	TMS	–	–
OScan4	nTDI	TMS	RDY	TDO		nTDI	–	RDY	TDO
OScan5	nTDI	TMS	–	TDO		nTDI	–	–	TDO
OScan6	–	TMS	–	–	Note 1	nTDI	TMS	–	TDO
					Note 2	nTDI	–	–	TDO
OScan7	–	TMS	–	–		nTDI	–	–	–

Note 1: Following Capture-xR or Exit2-xR Scan Packet

Note 2: Following Shift-xR Scan Packet

59.9 Functional Description

This section combines the information given in the Ancillary Services, EPU Operation, and APU Operation sections into a single reference for performing common TAP.7 tasks.

59.9.1 Switching from Standard Protocol to Advanced Protocol

Upon exit of Type-0 to Type-4 reset, the CJTAG begins operation in Standard Protocol 4-pin mode. The following steps are required to move from Standard Protocol 4-pin mode to Advanced Protocol (either 4-pin or 2-pin) mode.

The first step required to move to the Advanced Protocol is to change the CJTAG control level to control level 2 via zero-bit scans (ZBS), as described in Control levels.

After the control level is set tocontrol level 2, TAP.7 commands can be executed as described in EPU Commands. Execution of the Store Scan Format (STFMT) to set the scan format to any OSCAN0-7 or MSCAN format begins the switch from the Standard Protocol to the Advanced Protocol.

Once the STFMT command is executed selecting an Advanced Protocol scan format, the CJTAG transitions to the intermediate Control Protocol mode. The Control Protocol uses TMS inputs to control Configuration Change Packets as described in Control Process Active (CPA) Function and Configuration Change Packets (CP). Loading a change packet body value of 00 (CP_END) results in the switch to the Advanced Protocol.

If Advanced Protocol 2-pin operation is desired, the APFC register should be written to a value of 10 or 11 prior to the execution of the STFMT command that selects an advanced scan format. The APFC register is written via the Store Conditional 2-bit (STC2) command described in Store Conditional 2-bit (STC2).

Chapter 60
Nexus Port Controller (NPC)

60.1 Chip-specific NPC information

60.1.1 NPC configuration

This device contains two instances of the NPC module: NPC_0 and NPC_1. NPC_0 is responsible for the traces and interaction with pads and remains in full power domain. NPC_1 exists in always on domain and is used for low power handshaking with the debugger.

The NPC_PCR for NPC_0 is shown below.

Register index: 127

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FPM	MCKO_GT	MCKO_EN	MCKO_DIV			EVT_EN	Reserved	Reserved	0	0	0	Reserved		0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The NPC_PCR for NPC_1 is shown below.

Register index: 127

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LP_DBG	0	0	0	0	0	Reserved	LP1_SYN	0	0	0	0	0	0	0	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

60.1.2 EVTO sharing - Pulse time

In this device, whenever a Nexus client drives EVTO input of NPC for one S80 period, the NPC drives EVTO on pad for one S80 period on the following clock.

60.1.3 Trace messages for transactions with error responses

For transactions of trace clients which result in error responses, traces may or may not be generated. Such transactions will be reported to MEMU.

The Nexus clients in this device do not perform speculative execution. When speculative execution is done, a trace message may be output before it is known whether the branch was actually taken. Since the Nexus clients in this device do not support it, the CANCEL field is not present in the trace messages of this device.

60.1.4 Trace loss during resets

Whenever reset occurs (reset which breaks debugger trace connection) and nexus tracing is ON, debug trace data stored in internal FIFO corresponding to instructions executed before reset instruction will be lost.

60.1.5 EVTI_IN pin

EVTI_IN is an active low signal. The NPC Port Configuration Register's EVT_EN bit configures the EVTI_IN and EVTO_B pins for EVTI/EVTO port functionality. If it is chosen by the user to not use this configuration and to use the SIUL_MSCR configuration register for this purpose, then it is recommended that the user also sets the PUE and PUS bits to 1 along with setting the IBE and SSS configurations. Otherwise, a 0 event may be seen in EVTI_IN which will trigger the EVTI functions in chip.

60.2 Introduction

The following figure is a block diagram of the Nexus Port Controller (NPC) block.

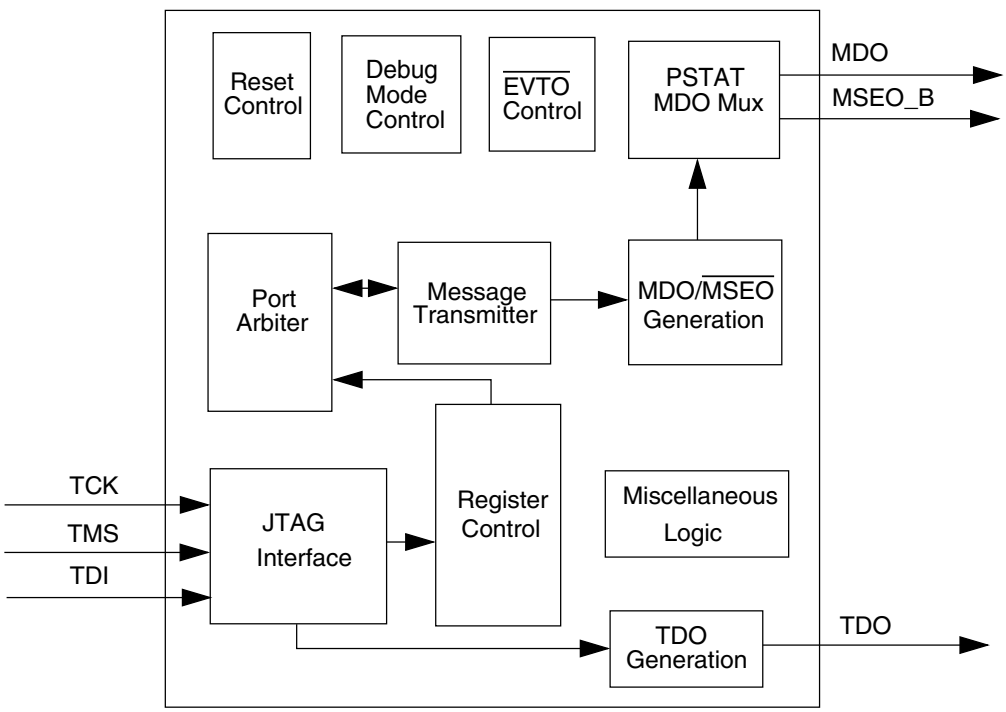


Figure 60-1. Nexus Port Controller Block Diagram

60.2.1 Overview

On a system-on-a-chip device, there are often multiple blocks that require development support. Each of these blocks implements a development interface based on the IEEE-ISTO 5001-2001 standard. The blocks share input and output ports that interface with the

development tool. The NPC controls the usage of the input and output port in a manner that allows all the individual development interface blocks to share the port, and appear to the development tool to be a single block.

60.2.2 Features

The NPC block performs the following functions:

- Controls arbitration for ownership of the Nexus Output Port
- Nexus Device Identification Register and Messaging
- Generates MCKO enable and frequency division control signals
- Controls sharing of $\overline{\text{EVTO}}$
- Generates an MCKO clock gating control signal to enable gating of MCKO when the auxiliary output port is idle
- Control of the device-wide debug mode
- Generates asynchronous reset signal for Nexus blocks based on censorship status, and power-on reset status
- System clock status indication via MDO[0] following power-on reset
- Provides Nexus support for censorship mode

60.2.3 Modes of operation

The NPC block uses the censorship status signal and an internal power-on reset indication as its primary reset signals. Upon exit of reset, the mode of operation is determined by the Port Configuration Register (PCR) settings.

60.2.3.1 Reset

The NPC block is asynchronously placed in reset when either power-on reset is asserted , the device enters censored mode , or the TAP controller state machine is in the Test-Logic-Reset state. Holding TMS high for 5 consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state.

Following negation of power-on reset, the NPC remains in reset until the system clock achieves lock. The NPC is unaffected by other sources of reset. While in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state
- The output port pins are negated
- The TDI, TMS, and TCK TAP inputs are ignored (when in power-on reset, censored mode
- Registers default back to their reset values

60.2.3.2 Disabled-Port Mode

In disabled-port mode, output pin port enable signals are negated, thereby disabling message transmission. Any debug feature that generates messages can not be used. The primary features available are class 1 features and read/write access to the registers. Class 1 features include the ability to trigger a breakpoint event indication through $\overline{\text{EVTO}}$.

60.2.3.3 Full-Port Mode

Full-port mode (FPM) is entered by asserting the MCKO_EN and FPM bits in the PCR. In All trace features are enabled or can be enabled by writing the configuration registers via the TAP. The number of MDO pins available is device-specific.

60.2.3.4 Reduced-Port Mode

Reduced-port mode (RPM) is entered by asserting the MCKO_EN bit and negating the FPM bit in the PCR. All trace features are enabled or can be enabled by writing the configuration registers via the TAP. The number of MDO pins available is device-specific.

60.2.3.5 Censored Mode

In censored mode, it should not be possible to read the contents of internal Flash externally. The NPC and all other Nexus blocks are held in reset when in censored mode , preventing Nexus read/write to memory mapped resources and the transmission of Nexus trace messages.

60.3 External signal description

60.3.1 Overview

The NPC pin interface provides for the transmission of messages from Nexus blocks to the external development tools and for access to Nexus client registers. The NPC pin definition is outlined in the following table. Pull information for the JTAG signals can be found in the JTAG chapter.

Table 60-3. NPC Signal Properties

Name	Port	Function	Reset State
EVTO_B	Nexus	Event Out pin	0b1
MDO	Auxiliary	Message Data Out pins	0[1]
MSEO	Auxiliary	Message Start/End Out pins	0b11
TCK	JTAG	Test Clock Input	—
TDI	JTAG	Test Data Input	—
TDO	JTAG	Test Data Output	High Z[2]
TMS	JTAG	Test Mode Select Input	—

1. Following a power-on reset, MDO[0] remains asserted until power-on reset is exited and the system clock achieves lock.
2. TDO output buffer enable is negated when the NPC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented on TDO at the SoC level.

60.3.2 Detailed signal descriptions

This section describes each of the signals listed in Table 60-3 in more detail. Note that the JTAG test clock (TCK) input from the pin is not a direct input to the NPC. The NPC requires two separate input clocks for TCK clocked logic, one for posedge (rising edge TCK) logic and one for negedge (falling edge TCK) logic. Both clocks are derived from the pin TCK, and generated external to the NPC.

60.3.2.1 EVTO_B — Event Out

Event Out ($\overline{\text{EVTO}}$) is an output pin that is asserted upon breakpoint occurrence to provide breakpoint status indication. The $\overline{\text{EVTO}}$ output of the NPC is generated based on the values of the individual $\overline{\text{EVTO}}$ signals from all Nexus blocks that implement the signal.

60.3.2.2 MDO — Message Data Out

Message Data Out (MDO) are output pins used for uploading OTM, BTM, DTM, and other messages to the development tool. The development tool should sample MDO on the rising edge of MCKO. The width of the MDO bus used is determined by reset configuration.

60.3.2.3 MSEO_B — Message Start/End Out

Message Start/End Out ($\overline{\text{MSEO}}$) is an output pin that indicates when a message on the MDO pins has started, when a variable length packet has ended, or when the message has ended. The development tool should sample $\overline{\text{MSEO}}$ on the rising edge of MCKO.

60.3.2.4 TCK — Test Clock Input

Test Clock Input (TCK) pin is used to synchronize the test logic and control register access through the JTAG port.

60.3.2.5 TDI — Test Data Input

Test Data Input (TDI) pin receives serial test instruction and data. TDI is sampled on the rising edge of TCK.

60.3.2.6 TDO — Nexus Test Data Output

Test Data Output (TDO) pin transmits serial output for instructions and data. TDO is three-stateable and is actively driven in the SHIFT-IR and SHIFT-DR controller states. TDO is updated on the falling edge of TCK and sampled by the development tool on the rising edge of TCK.

60.3.2.7 TMS — Test Mode Select

Test Mode Select Input (TMS) pin is used to sequence the IEEE 1149.1-2001 TAP controller state machine. TMS is sampled on the rising edge of TCK.

60.4 Register definition

This section provides a detailed description of the NPC registers accessible to the end user. Individual bit-level descriptions and reset states of the registers are included.

The following table shows the NPC registers by index values. The registers are not memory-mapped and can only be accessed via the TAP. The NPC block does not implement the client select control register because the value does not matter when accessing the registers.

Note that the bypass and instruction registers have no index values. These registers are not accessed in the same manner as Nexus client registers. Refer to the individual register descriptions for more detail.

Table 60-4. NPC registers

Index	Register
0	Device ID Register (DID)
127	Port Configuration Register (PCR)

60.4.1 Register descriptions

This section consists of NPC register descriptions.

60.4.1.1 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the BYPASS instruction or any unimplemented instructions are active. After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

60.4.1.2 Instruction register

The NPC block uses a 4-bit instruction register as shown in the following table. The instruction register is accessed via the SELECT_IR_SCAN path of the tap controller state machine, and allows instructions to be loaded into the block to enable the NPC for register access (NEXUS_ENABLE) or select the bypass register as the shift path from TDI to TDO (BYPASS or unimplemented instructions).

Instructions are shifted in through TDI while the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states. Synchronous entry into the Test-Logic-Reset state results in synchronous loading of the BYPASS instruction. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the BYPASS instruction. During the Capture-IR TAP controller state, the instruction register is loaded with the value of the previously executed instruction, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

	3	2	1	0
R	Previous Instruction Opcode			
W	Instruction Opcode			
Reset:	BYPASS Instruction Opcode (0xF)			

60.4.1.3 Nexus Device ID (DID) Register

The device identification register, shown in the following figure, allows the part revision number, design center, part identification number, and manufacturer identity code of the part to be determined through the output port.

Register index: 0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Part Revision Number				Design Center						Part Identification Number					
W																
Reset	PRN				DC						PIN					

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Part Identification Number				Manufacturer Identity Code											1
W																
Reset	PIN (cont'd)				0	0	0	0	0	0	0	1	1	1	0	1

Table 60-7. DID field descriptions

Bit	Name	Description
31:28	PRN	Part Revision Number These bits contain the revision number of the part
27:22	DC	Design Center These bits indicate the device design center
21:12	PIN	Part Identification Number These bits contain the part number of the device
11:1	MIC	Manufacturer Identity Code These bits contain the reduced Joint Electron Device Engineering Council (JEDEC) ID for NXP, 0xE.
0	Bit [0]	IDCODE Register ID Bit [0] identifies this register as the device identification register and not the bypass register

60.4.1.4 Port Configuration Register (PCR)

Register index: 127

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FPM	MCKO_GT	MCKO_EN	MCKO_DIV			EVT_EN	Reserved	Reserved	0	0	0	Reserved		0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	0	0	0	0	0	Reserved	Reserved	0	0	0	0	0	0	0	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The PCR, shown in the figure above, is used to select the NPC mode of operation, enable MCKO and select the MCKO frequency, and enable or disable MCKO gating. This register should be configured as soon as the NPC is enabled.

NOTE

The mode or clock division must not be modified after MCKO has been enabled. Changing the mode or clock division while MCKO is enabled can produce unpredictable results.

Table 60-9. PCR field descriptions

Bit	Name	Description
31	FPM	Full Port Mode The value of the FPM bit determines if the auxiliary output port uses the full MDO port or a reduced MDO port to transmit messages. 0 A subset of MDO pins are used to transmit messages 1 All MDO pins are used to transmit messages
30	MCKO_GT	MCKO Clock Gating Control This bit is used to enable or disable MCKO clock gating. If clock gating is enabled, the MCKO clock is gated when the NPC is in enabled mode but not actively transmitting messages on the auxiliary output port. When clock gating is disabled, MCKO is allowed to run even if no auxiliary output port messages are being transmitted. 0 MCKO gating is disabled 1 MCKO gating is enabled
29	MCKO_EN	MCKO Enable This bit enables the MCKO clock to run. When enabled, the frequency of MCKO is determined by the MCKO_DIV field. 0 MCKO clock is driven to zero 1 MCKO clock is enabled
28:26	MCKO_DIV	MCKO Division Factor The value of this signal determines the frequency of MCKO relative to the system clock frequency when MCKO_EN is asserted. Table 60-10 shows the meaning of MCKO_DIV Values. In this table, SYS_CLK represents the system clock frequency.
25	EVT_EN	EVTO/EVTI Enable This bit enables the EVTO/EVTI port functions. 0 EVTO/EVTI port disabled 1 EVTO/EVTI port enabled
24	Reserved	
23	Reserved	
19:18	Reserved	
15	Reserved	
9	Reserved	
8	Reserved	
0	Reserved	

Table 60-10. MCKO_DIV Values

MCKO_DIV[2:0]	MCKO Frequency
0	Reserved
1	SYS_CLK/2
2	Reserved
3	SYS_CLK/4

Table continues on the next page...

Table 60-10. MCKO_DIV Values (continued)

MCKO_DIV[2:0]	MCKO Frequency
4	Reserved
5	Reserved
6	Reserved
7	SYS_CLK/8

60.5 Functional Description

60.5.1 NPC reset configuration

The NPC is placed in disabled mode upon exit of reset. If message transmission via the output port is desired, a write to the PCR is then required to enable the NPC and select the mode of operation. Asserting MCKO_EN places the NPC in enabled mode and enables MCKO. The frequency of MCKO is selected by writing the MCKO_DIV field. Asserting or negating the FPM bit selects full-port or reduced-port mode, respectively.

The following table describes the NPC reset configuration options.

Table 60-11. NPC reset configuration options

Power On Reset status asserted?	MCKO_EN bit of the Port Configuration Register	FPM bit of the Port Configuration Register	Configuration
yes	X	X	Reset
no	0	X	Disabled
no	1	1	Full-Port Mode
no	1	0	Reduced-Port Mode

60.5.2 Auxiliary Output Port

The auxiliary output port is shared by each of the Nexus modules on the device. The NPC communicates with each of the Nexus modules and arbitrates for access to the port.

60.5.2.1 Output Message Protocol

The protocol for transmitting messages via the auxiliary port is accomplished with the $\overline{\text{MSEO}}$ functions. The $\overline{\text{MSEO}}$ pins are used to signal the end of variable-length packets and the end of messages. They are not required to indicate the end of fixed-length packets. MDO and $\overline{\text{MSEO}}$ are sampled on the rising edge of MCKO.

The following figure illustrates the state diagram for $\overline{\text{MSEO}}$ transfers. All transitions not included in the figure are reserved, and must not be used.

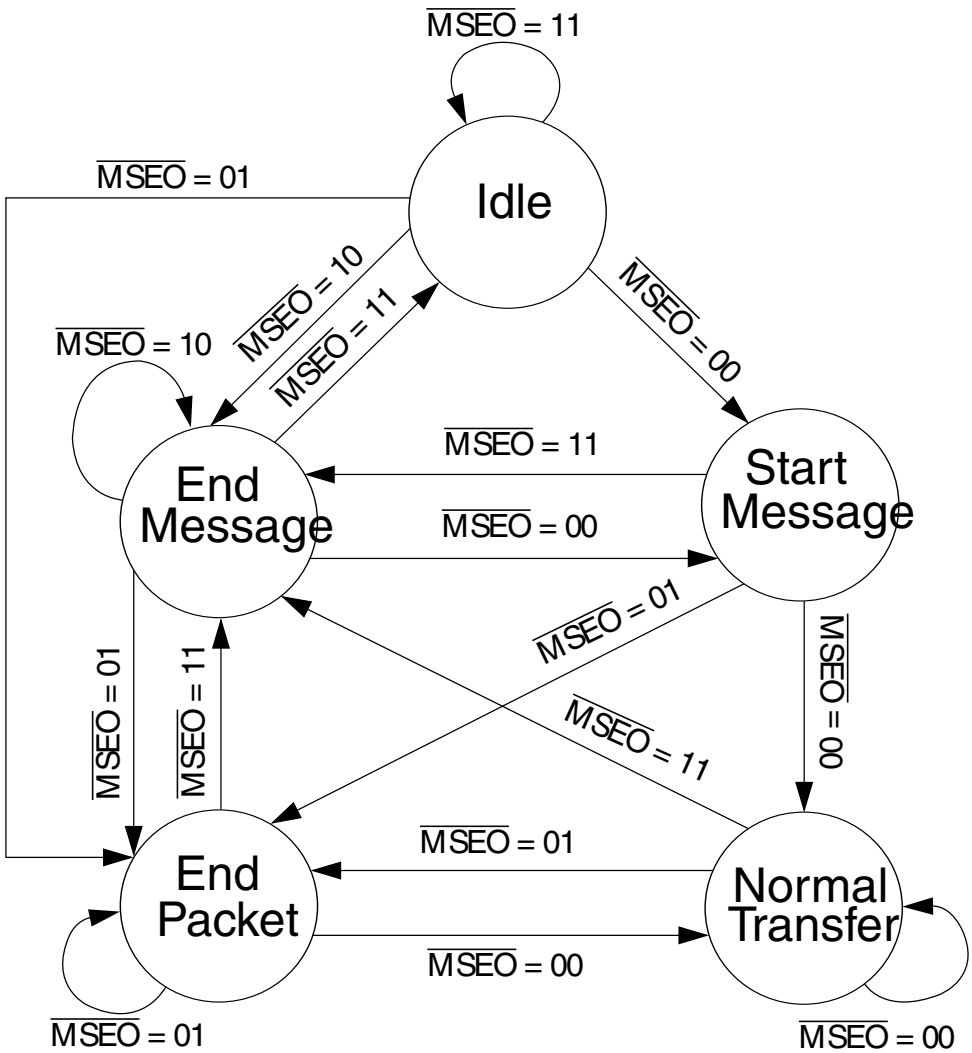


Figure 60-2. $\overline{\text{MSEO}}$ Transfers (for 2-bit $\overline{\text{MSEO}}$)

60.5.2.2 Output Messages

In addition to sending out messages generated in other Nexus blocks, the NPC can also output the device ID message contained in the device ID register on the MDO pins. The device ID message can also be sent out serially through TDO.

Figure 60-3 describes the device ID output message that the NPC can transmit on the auxiliary port. The TCODE is the first packet transmitted.

Message Name	Min. Packet Size (bits)	Max Packet Size (bits)	Packet Type	Packet Name	Packet Description
Device ID Message	6	6	fixed	TCODE	Value = 1
	32	32	fixed	ID	DID register contents

Figure 60-3. NPC output messages

60.5.2.3 Rules of Message

- A variable-sized field within a message must end on a port boundary. (Port boundaries depend on the number of MDO pins active with the current reset configuration.)
- A variable-sized field may start within a port boundary only when following a fixed-length field.
- Super-fields must end on a port boundary.
- When a variable-length field is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest order bit so that it can end on a port boundary.
- Multiple fixed-length packets may start and/or end on a single clock.
- When any packet follows a variable-length packet, it must start on a port boundary.
- The field containing the TCODE number is always transferred out first, followed by subsequent fields of information.
- Within a field, the lowest significant bits are shifted out first. The following figure shows the transmission sequence of a message that is made up of a TCODE followed by two fields.

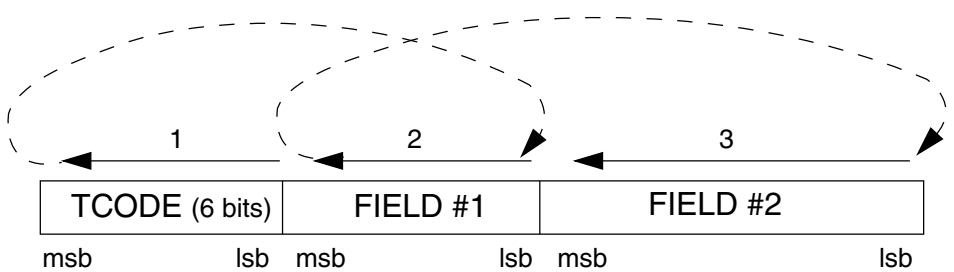


Figure 60-4. Transmission Sequence of Messages

60.5.3 IEEE 1149.1-2001 (JTAG) TAP

The NPC block uses the IEEE 1149.1-2001 test access port (TAP) for accessing registers. Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers as well. TAP signals include TCK, TDI, TMS, and TDO. There may also be other blocks on the MCU that use the TAP and implement a TAP controller.

Refer to the IEEE 1149.1-2001 specification for further detail on electrical and pin protocol compliance requirements.

The NPC implements a Nexus controller state machine that transitions based on the state of the IEEE 1149.1-2001 state machine shown in Figure 60-6 . The Nexus controller state machine is defined by the IEEE-ISTO 5001-2001 standard. It is shown in Figure 60-7.

The instructions implemented by the NPC TAP controller are listed in the following table. The value of the NEXUS-ENABLE instruction is 0b0000. Each unimplemented instruction acts like the BYPASS instruction. The size of the NPC instruction register is 4 bits.

Table 60-12. Implemented Instructions

Instruction Name	Private/Public	Opcode	Description
NEXUS-ENABLE	public	0x0	Activate Nexus controller state machine to read and write NPC registers.
BYPASS	private	0xF	NPC BYPASS instruction. Also the value loaded into the NPC IR upon exit of reset.

Data is shifted between TDI and TDO starting with the least significant bit, as illustrated in the following figure. This applies for the instruction register and all Nexus tool-mapped registers.

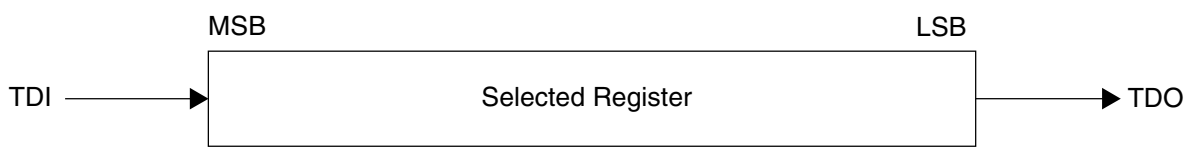
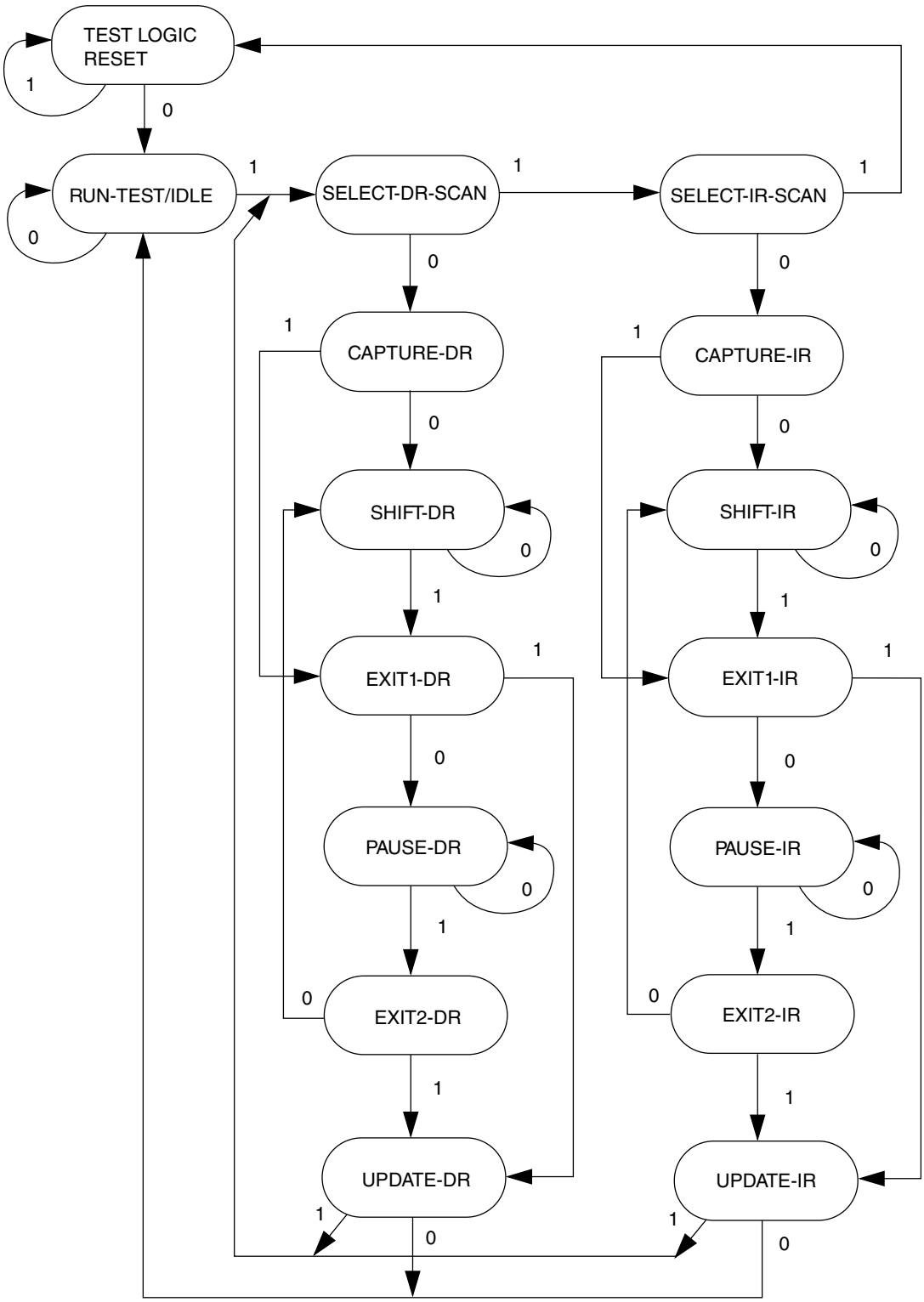


Figure 60-5. Shifting Data Into Register

60.5.3.1 Enabling the NPC TAP controller

Assertion of the power-on reset signal or entry into censored mode resets the NPC TAP controller. When not in power-on reset or censored mode, the NPC TAP controller is enabled by exiting the Test-Logic-Reset state. Loading the NEXUS-ENABLE instruction then grants access to Nexus debug.



NOTE: The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

Figure 60-6. IEEE 1149.1-2001 TAP Controller State Machine

60.5.3.2 Retrieving device IDCODE

The Nexus TAP controller does not implement the IDCODE instruction. However, the device identification message can be output by the NPC through the output port or shifted out serially by accessing the Nexus Device ID register through the TAP.

Transmission of the device identification message on the auxiliary output port MDO pins occurs immediately after a write to the PCR, if the NPC is enabled for auxiliary port mode. Transmission of the device identification message serially via the TAP is achieved by performing a read of the register contents as described in Selecting a Nexus Client Register.

60.5.3.3 Loading NEXUS-ENABLE Instruction

Access to the NPC registers is enabled when the TAP controller instruction register is loaded with the NEXUS-ENABLE instruction. This instruction is shifted in via the SELECT-IR-SCAN path and loaded in the UPDATE-IR state. At this point, the Nexus controller state machine, shown in Figure 60-7, transitions to the REG_SELECT state. The Nexus controller has three states: idle, register select, and data access. Table 60-13 illustrates the IEEE 1149.1 sequence to load the NEXUS-ENABLE instruction.

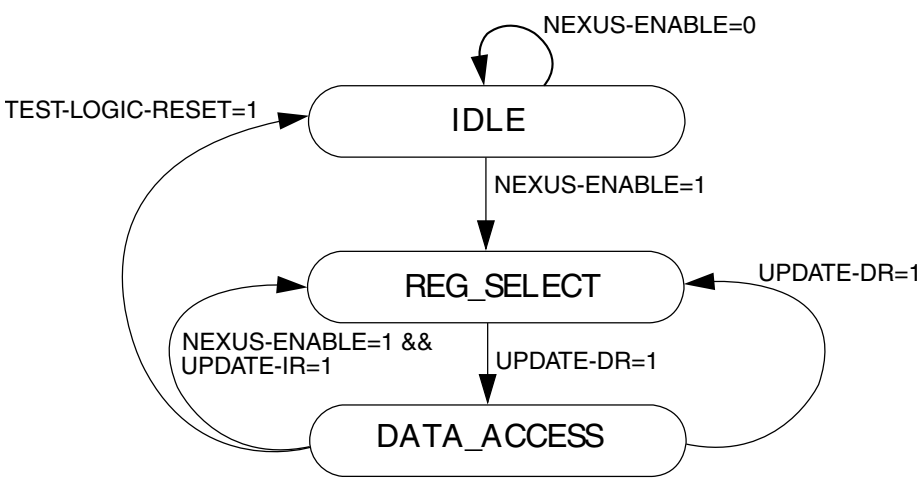


Figure 60-7. NEXUS Controller State Machine

Table 60-13. Loading NEXUS-ENABLE instruction

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	IDLE	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	IDLE	Transitional state
2	1	SELECT-IR-SCAN	IDLE	Transitional state
3	0	CAPTURE-IR	IDLE	Internal shifter loaded with current instruction

Table continues on the next page...

Table 60-13. Loading NEXUS-ENABLE instruction (continued)

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
4	0	SHIFT-IR	IDLE	TDO becomes active, and the IEEE 1149.1-2001 shifter is ready. Shift in all but the last bit of the NEXUS_ENABLE instruction.
3 TCKS				
8	1	EXIT1-IR	IDLE	Last bit of instruction shifted in
9	1	UPDATE-IR	IDLE	NEXUS-ENABLE loaded into instruction register
10	0	RUN-TEST/IDLE	REG_SELECT	Ready to be read/write Nexus registers

60.5.3.4 Selecting a Nexus Client Register

When the NEXUS-ENABLE instruction is decoded by the TAP controller, the input port allows development tool access to all Nexus registers. Each register has a 7-bit address index.

All register access is performed via the SELECT-DR-SCAN path. The Nexus Controller defaults to the REG_SELECT state when enabled. Accessing a register requires two passes through the SELECT-DR-SCAN path: one pass to select the register and the second pass to read/write the register.

The first pass through the SELECT-DR-SCAN path is used to enter an 8-bit Nexus command consisting of a read/write control bit in the LSB followed by a 7-bit register address index, as illustrated in the following table. The read/write control bit is set to 1 for writes and 0 for reads.

MSB	LSB
7-bit register index	R/W

Figure 60-8. IEEE 1149.1 Controller Command Input

The second pass through the SELECT-DR-SCAN path is used to read or write the register data by shifting in the data (LSB first) during the SHIFT-DR state. When reading a register, the register value is loaded into the JTAG shifter during the CAPTURE-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the UPDATE-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated once the required number of bits have been acquired.

The following table illustrates a sequence which writes a 32-bit value to a register.

Table 60-14. Write to a 32-Bit Nexus Client Register

Clock	TMS	IEEE 1149.1 State	Nexus State	Description
0	0	RUN-TEST/IDLE	REG_SELECT	IEEE 1149.1-2001 TAP controller in idle state
1	1	SELECT-DR-SCAN	REG_SELECT	First pass through SELECT-DR-SCAN path
2	0	CAPTURE-DR	REG_SELECT	Internal shifter loaded with current value of controller command input.
3	0	SHIFT-DR	REG_SELECT	TDO becomes active, and write bit and 6 bits of register index shifted in.
7 TCKs				
11	1	EXIT1-DR	REG_SELECT	Last bit of register index shifted into TDI
12	1	UPDATE-DR	REG_SELECT	Controller decodes and selects register
13	1	SELECT-DR-SCAN	DATA_ACCESS	Second pass through SELECT-DR-SCAN path
14	0	CAPTURE-DR	DATA_ACCESS	Internal shifter loaded with current value of register
15	0	SHIFT-DR	DATA_ACCESS	TDO becomes active, and outputs current value of register while new value is shifted in through TDI
31 TCKs				
47	1	EXIT1-DR	DATA_ACCESS	Last bit of current value shifted out TDO. Last bit of new value shifted in TDI.
48	1	UPDATE-DR	DATA_ACCESS	Value written to register
49	0	RUN-TEST/IDLE	REG_SELECT	Controller returned to idle state. It could also return to SELECT-DR-SCAN to write another register.

60.5.4 Nexus JTAG Port Sharing

Each of the individual Nexus blocks on the device implements a TAP controller for accessing its registers. When Nexus has ownership of the TAP, only the block whose NEXUS-ENABLE instruction is loaded has control of the TAP. This allows the interface to all of these individual TAP controllers to appear to be a single port from outside the device. If no register is selected as the shift path for a Nexus block, that block acts like a single-bit shift register, or bypass register.

60.5.5 MCKO Control

MCKO is an output clock to the development tools used for the timing of $\overline{\text{MSEO}}$ and MDO pin functions. MCKO is derived from the system clock and its frequency is determined by the value of the MCKO_DIV field in the PCR. Possible operating frequencies include system clock, one-half system clock, one-quarter system clock, and one-eighth system clock speed.

The NPC also generates an MCKO clock gating control output signal. This output can be used by the MCKO generation logic to gate the transmission of MCKO when the auxiliary port is enabled but not transmitting messages. The setting of the MCKO_GT bit inside the PCR determines whether or not MCKO gating control is active. The MCKO_GT bit resets to a logic 0. In this state gating of MCKO is disabled. To enable gating of MCKO, the MCKO_GT bit in the PCR is written to a logic 1.

60.5.6 $\overline{\text{EVTO}}$ Sharing

The NPC block controls sharing of the $\overline{\text{EVTO}}$ output between all Nexus clients that produce an $\overline{\text{EVTO}}$ signal. The NPC assumes incoming $\overline{\text{EVTO}}$ signals will be asserted for one system clock period. After receiving a single clock period of asserted $\overline{\text{EVTO}}$ from any Nexus client, the NPC latches the result, and drives $\overline{\text{EVTO}}$ for one MCKO period on the following clock. When there is no active MCKO, such as in disabled mode, the NPC drives $\overline{\text{EVTO}}$ for two system clock periods. $\overline{\text{EVTO}}$ sharing is active as long as the NPC is not in reset.

60.5.7 System Clock Locked Indication

Following a power-on reset, MDO[0] can be monitored to provide the lock status of the system clock. MDO[0] is driven to a logic 1 until the system clock achieves lock after exiting power-on reset. Once the system clock is locked, MDO[0] is negated and tools may begin Nexus configuration. Loss of lock conditions that occur subsequent to the exit of power-on reset and the initial lock of the system clock do not cause a Nexus reset, and therefore do not result in MDO[0] driven high.

60.6 Initialization/Application Information

60.6.1 Accessing NPC tool-mapped registers

To initialize the TAP for Nexus register accesses, the following sequence is required:

1. Enable the Nexus TAP controller
2. Load the TAP controller with the NEXUS-ENABLE instruction

To write control data to NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and set the write bit to select the register with a pass through the SELECT-DR-SCAN path in the TAP controller state machine.
2. Write the register value with a second pass through the SELECT-DR-SCAN path. Note that the prior value of this register is shifted out during the write.

To read status and control data from NPC tool-mapped registers, the following sequence is required:

1. Write the 7-bit register index and clear the write bit to select register with a pass through SELECT-DR-SCAN path in the TAP controller state machine.
2. Read the register value with a second pass through the SELECT-DR-SCAN path. Data shifted in is ignored.

See the IEEE-ISTO 5001-2001 standard for more detail.

Chapter 61
Core Complex Overview

61.1 Overview of the e200z210n3 and e200z4204n3 cores

The e200z processor family is a set of CPU cores that implement low-cost versions of the Power architecture.

This chapter briefly discusses the e200z210n3 and e200z4204n3 cores as implemented on this device. Separate chapters discuss the processing models, debug capabilities, and Nexus 3+ units for each core.

This device uses one e200z4204n3 core (z4204n3) operating at speeds as high as 160 MHz, and one e200z210n3 core (z210n3) operating at a speed as high as 80 MHz.

All cores on the device implement a 32-bit PowerISA 2.06 VLE-compliant design with 32-bit general purpose registers (GPRs). The VLE (variable-length encoding) ISA provides improved code density. The VLE ISA is further documented in *PowerISA 2.06*, a separate document.

Note

The z4204n3 core is a dual-issue core. The z210n3 core is a single-issue core.

61.2 z210n3 microarchitecture summary

The z210n3 core utilizes a four-stage instruction pipeline, with one stage for execution. The Instruction Fetch, Instruction Decode/Register file Read/ EA Calc, Execute / Memory Access, and Register Writeback stages operate in an overlapped fashion, allowing single clock instruction execution for most instructions.

The integer execution unit consists of a 32-bit Arithmetic Unit (AU), a Logic Unit (LU), a 32-bit Barrel shifter (Shifter), a Mask-Insertion Unit (MIU), a Condition Register manipulation Unit (CRU), a Count-Leading-Zeros unit (CLZ), and result feed-forward hardware. The Integer EU1 also supports hardware division and an 8x32 Hardware Multiplier array.

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

The Instruction Unit contains a PC incrementer and dedicated Branch Address adders to minimize delays during change of flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching is performed to accelerate taken branches. Prefetched instructions are placed into an instruction buffer.

Branch target addresses are calculated in parallel with branch instruction decode, resulting in execution time of two clocks for taken branches. Conditional branches which are not taken execute in a single clock. Branches with successful BTB target prefetching have an effective execution time of one clock if correctly predicted.

Memory load and store operations are provided for byte, halfword, and word (32-bit) data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal of data. These instructions are pipelined to allow effective single cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations, and utilize 64-bit data transfers when possible. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized.

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

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

61.2.1 Overview of the e200z210n3 core

The e200z210n3 core (z210n3) is a single-issue 32-bit PowerISA 2.06 VLE-compliant design with 32-bit general purpose registers (GPRs). The z210n3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *PowerISA 2.06*, a separate document.

The z210n3 core integrates an integer execution unit, a branch control unit, an instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining three read and two write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

The z210n3 core contains a Nexus Class 3+ real-time debug module with support for program and data trace features and extensive trace controls.

Note

Instruction and data cache are not implemented on the z210n3 core.

The following lists some of the key features of the z210n3 core:

- Single-issue 32-bit PowerISA 2.06-VLE compliant CPU
- In-order execution and retirement
- Precise exception handling
- Branch processing unit
  - Dedicated branch address calculation adder
  - Branch target prefetching using BTB
- Load/store unit
  - Single-cycle load latency
  - Misaligned access support
- 32-bit General Purpose Register file
- Dual AHB 2.v6 64-bit Instruction-side and Data-side System buses
- Performance Monitor APU supporting execution profiling
- Nexus Class 3+ Real-time Development Unit
- Power management
  - Low power design - extensive clock gating
  - Power saving modes: halt, stop, wait
  - Dynamic power management of execution units
- Testability
  - Synthesizeable, MuxD scan design

61.2.2 z210n3 instruction unit features

The features of the z210n3 instruction unit are:

- Support for fetching two 32-bit instruction per clock cycle
- Instruction buffer that holds as many as eight 32-bit instructions
- Dedicated PC incrementer that supports instruction prefetches
- Branch unit with dedicated branch address adder and branch lookahead logic (BTB) to support single-cycle execution of successfully predicted branches
- Saturation Instructions Extension to provide scalar saturating arithmetic support to the PowerISA. Instructions supports add and subtract functionality for both signed and unsigned word, halfword, and byte data types. These instructions are used by compilers to avoid sign extension or zero extension operations when operating on data types which are smaller than the implemented register size.

61.2.3 z210n3 integer unit features

The z210n3 integer unit supports single-cycle execution of most integer instructions:

- 32-bit AU for arithmetic and comparison operations
- 32-bit LU for logical operations
- 32-bit priority encoder for the count leading zero's function
- 32-bit single-cycle barrel shifter for static shifts and rotates
- 32-bit mask unit for data masking and insertion
- Divider logic for signed and unsigned divide in 5-34 clock cycles with minimized execution timing
- Pipelined 8 x 32 hardware multiplier array supports 32x32->32 multiply with 2-4 clock cycle latency

61.2.4 z210n3 load/store unit features

The z210n3 load/store units support load, store, and the load multiple / store multiple instructions:

- 32-bit effective address adder for data memory address calculations
- Dedicated 64-bit interface to memory supports saving and restoring of as many as two registers per clock cycle for load multiple and store multiple word instructions and context save/restore instructions
- Pipelined operation to support throughput of one load or store operation per clock cycle

61.2.5 z210n3 performance monitor features

The z210n3 performance monitor provides the following features:

- Four configurable 32-bit performance monitor counters, each capable of counting selected CPU subsystem events
- Performance Monitor interrupt
- Ability to configure performance monitor resources for debugger use
- Hardware input signals for qualification of counting by individual counters
- Hardware output signals to indicate counter overflows
- Dedicated watchpoint outputs for each counter with programmable periodicity
- Trigger On/Off control for each counter based on subsystem events

Note

Since the z210n3 core does not implement caches, the available selectable event types represent a subset of the selectable event types on the z4204n3 core.

61.2.6 z210n3 system bus features

The features of the z210n3 system bus interface are as follows:

- Independent instruction and data interfaces
- Advanced microcontroller bus architecture AMBA AHB2.v6 protocol
- 32-bit address bus, 64-bit instruction-side and data-side system buses
- Data interface provides separate uni-directional read and write data buses
- Support for HCLK running at a slower rate than CPU clock

Note

These are identical to the z4204n3 system bus features.

61.3 z4204n3 microarchitecture summary

Each z4204n3 core utilizes a five-stage instruction pipeline, with two stages for execution.

Note

The z210n3 core utilizes a four-stage instruction pipeline, with one stage for execution.

The Instruction Fetch, Instruction Decode/Register file Read/ EA Calc, Execute 0/ Memory Access 0, Execute1/Memory Access1, and Register Writeback stages operate in an overlapped fashion, allowing single clock instruction execution for most instructions.

The integer execution units each consists of a 32-bit Arithmetic Unit (AU), a Logic Unit (LU), a 32-bit Barrel shifter (Shifter), a Mask-Insertion Unit (MIU), a Condition Register manipulation Unit (CRU), a Count-Leading-Zeros unit (CLZ), and result feed-forward hardware. The Integer EU1 also supports hardware division and a 32x32 Hardware Multiplier array.

Most arithmetic and logical operations are executed in a single cycle with the exception of multiply, which is implemented with a pipelined hardware array, and the divide instructions. A Count-Leading-Zeros unit operates in a single clock cycle.

The Instruction Unit contains a PC incrementer and dedicated Branch Address adders to minimize delays during change of flow operations. Sequential prefetching is performed to ensure a supply of instructions into the execution pipeline. Branch target prefetching is performed to accelerate taken branches. Prefetched instructions are placed into an instruction buffer.

Branch target addresses are calculated in parallel with branch instruction decode, resulting in execution time of two clocks for correctly predicted taken branches. Conditional branches that are not taken will execute in a single clock. Branches with successful BTB target prefetching have an effective execution time of one clock if correctly predicted.

Memory load and store operations are provided for byte, halfword, and word (32-bit) data with automatic zero or sign extension of byte and halfword load data as well as optional byte reversal of data. These instructions are pipelined to allow effective single cycle throughput. Load and store multiple word instructions allow low overhead context save and restore operations, and utilize 64-bit data transfers when possible. The load/store unit contains a dedicated effective address adder to allow effective address generation to be optimized.

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

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

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

61.3.1 Overview of the e200z4204n3

The e200z4204n3 core (z4204n3) is a dual-issue 32-bit PowerISA 2.06 VLE compliant design with 32-bit general purpose registers (GPRs). The z4204n3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in PowerISA 2.06, a separate document.

Embedded Floating-point (EFPU2) APUs in each the z4204n3 cores support real-time single-precision floating-point embedded numerics operations using the general-purpose registers. Each z4204n3 core integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

The z4204n3 cores each contain an 8 KB Instruction cache and a 4 KB Data cache, and a Nexus Class 3+ real-time debug module with support for program and data trace features and extensive trace controls.

61.3.2 z4204n3 instruction unit features

The features of the z4204n3 instruction units are:

- Support for fetching two 32-bit instruction per clock cycle
- Instruction buffer that holds as many as eight 32-bit instructions
- Dedicated PC incrementer that supports instruction prefetches

- Branch unit with dedicated branch address adder and branch lookahead logic (BTB) to support single-cycle execution of successfully predicted branches
- Saturation Instructions Extension to provide scalar saturating arithmetic support to the PowerISA. Instructions supports add and subtract functionality for both signed and unsigned word, halfword, and byte data types. These instructions are used by compilers to avoid sign extension or zero extension operations when operating on data types which are smaller than the implemented register size.

61.3.3 z4204n3 integer unit features

The z4204n3 integer units support single-cycle execution of most integer instructions:

- 32-bit AU for arithmetic and comparison operations
- 32-bit LU for logical operations
- 32-bit priority encoder for the count leading zero's function
- 32-bit single-cycle barrel shifter for static shifts and rotates
- 32-bit mask unit for data masking and insertion
- Divider logic for signed and unsigned divide in 4-14 clocks with minimized execution timing (EU1 only)
- Pipelined 32x32 hardware multiplier array supports 32x32->32 multiply with 2 clock latency, 1 clock throughput (EU1 only)

61.3.4 z210n3 load/store unit features

The z210n3 load/store units support load, store, and the load multiple / store multiple instructions:

- 32-bit effective address adder for data memory address calculations
- Dedicated 64-bit interface to memory supports saving and restoring of as many as two registers per clock cycle for load multiple and store multiple word instructions and context save/restore instructions
- Pipelined operation to support throughput of one load or store operation per clock cycle

61.3.5 z4204n3 EFPU2 floating-point unit features

The z4204n3 embedded floating-point unit (EFPU2) supports pipelined operation of most floating-point operations, allowing a throughput of one floating-point operation per cycle, and supports a variety of operations:

- Supports IEEE754 single-precision data format
- Supports IEEE754 half-precision format for conversion two and from single-precision format
- EFPU2 instructions utilize the 32-bit GPRs for minimized context save/restore overhead
- Provides low latency pipelined scalar floating-point add, subtract, add/sub, sub/add, multiply, multiply-add, min, max, absolute value, compare, and conversion instructions with single-cycle throughput
- Default results mode for real-time applications allows for exception-free operations when underflow or overflows occur
- Trap Enable controls allow for trapping of various operation exceptions and emulation of IEEE754 boundary case results
- Floating-point Divide and Square root logic operating in 13 (FP div) and 11 (FP sqrt) clocks

Note

The EFPU2 is not implemented on the z210n3 core.

61.3.6 z4204n3 cache features

The z4204n3 caches provide the following features:

- 2-way set-associative Instruction and Data caches (8 KB ICache, 4 KB DCache)
- Writethrough support
- 8-entry store buffer
- Linefill buffer
- 32-bit address bus plus attributes and control
- Separate uni-directional 64-bit read data bus and 64-bit write data bus
- Support for way locking

- Support for Write allocation policies
- Support for multi-bit EDC with Correction/Auto-invalidation capability
- Hardware debug cache invalidate support for the data cache

Note

Instruction and data caches are not implemented on the z210n3 core.

61.3.7 z4204n3 performance monitor features

The z4204n3 performance monitor provides the following features:

- Four configurable 32-bit performance monitor counters, each capable of counting selected CPU subsystem events
- Performance Monitor interrupt
- Ability to configure performance monitor resources for debugger use
- Hardware input signals for qualification of counting by individual counters
- Hardware output signals to indicate counter overflows
- Dedicated watchpoint outputs for each counter with programmable periodicity
- Trigger On/Off control for each counter based on subsystem events
- Multiple selectable event types including:
  - Number of clocks, CPU stall cycles
  - Number of instructions completed
  - Number of interrupts taken
  - Number of memory accesses (by type)
  - Number of instruction or data cache misses
  - Number of cycles with 0, 1, or 2 instructions issued
  - Number of instructions in a given class completed (ldst, branch, integer, FP, etc.)
  - Number cache linefills
  - Many other event types

Note

For the z4204n3 core, selectable event types represent a superset of the selectable event types on the z210n3 core, since the z210n3 core does not have caches.

61.3.8 z4204n3 system bus features

The features of the z4204n3 system bus interface are as follows:

- Independent instruction and data interfaces
- Advanced microcontroller bus architecture AMBA AHB2.v6 protocol
- 32-bit address bus, 64-bit instruction-side and data-side system buses
- Data interface provides separate uni-directional read and write data buses
- Support for HCLK running at a slower rate than CPU clock

Note

These are identical to the z210n3 system bus features.

Chapter 62
Z2 Core description

62.1 Overview of the e200z210n3 core

The e200z210n3 is a single-issue 32-bit *Power ISA 2.06* VLE compliant design with 32-bit general-purpose registers (GPRs). The e200z210n3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *Power ISA 2.06*, a separate document.

The e200z210n3 core integrates an integer execution unit, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining three read and two write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

The e200z210n3 core contains a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

62.2 Register model

This section describes the registers implemented in the e200z210n3 core. The *Power ISA 2.06* architecture defines register-to-register operations for all computational instructions.

Figure 62-1 and Figure 62-2 show the complete e200z210n3 register set, all of which are accessible in supervisor mode. Figure 62-3 and Figure 62-4 show the subset of registers accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

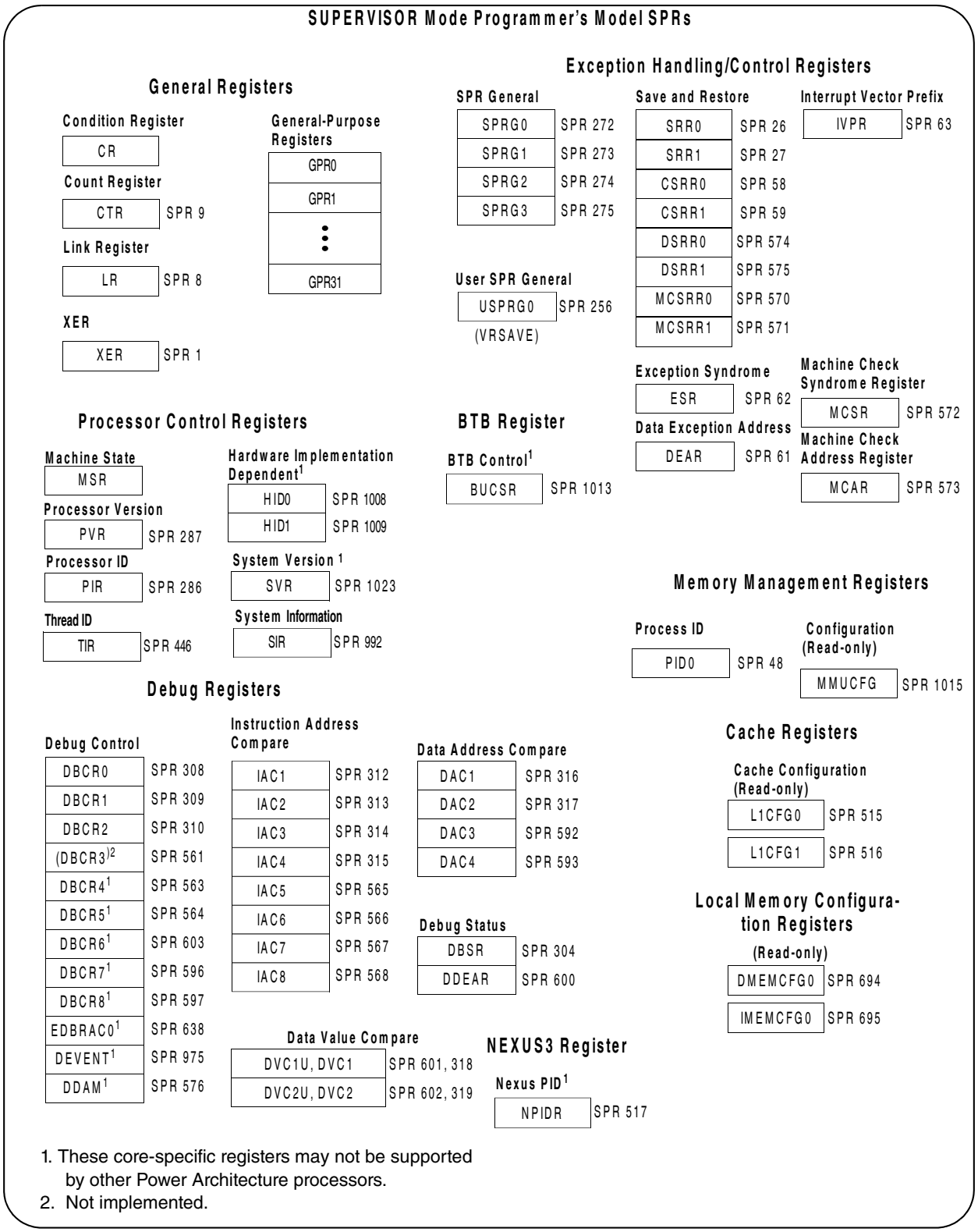


Figure 62-1. e200z210n3 Supervisor Mode Programmer's Model SPRs

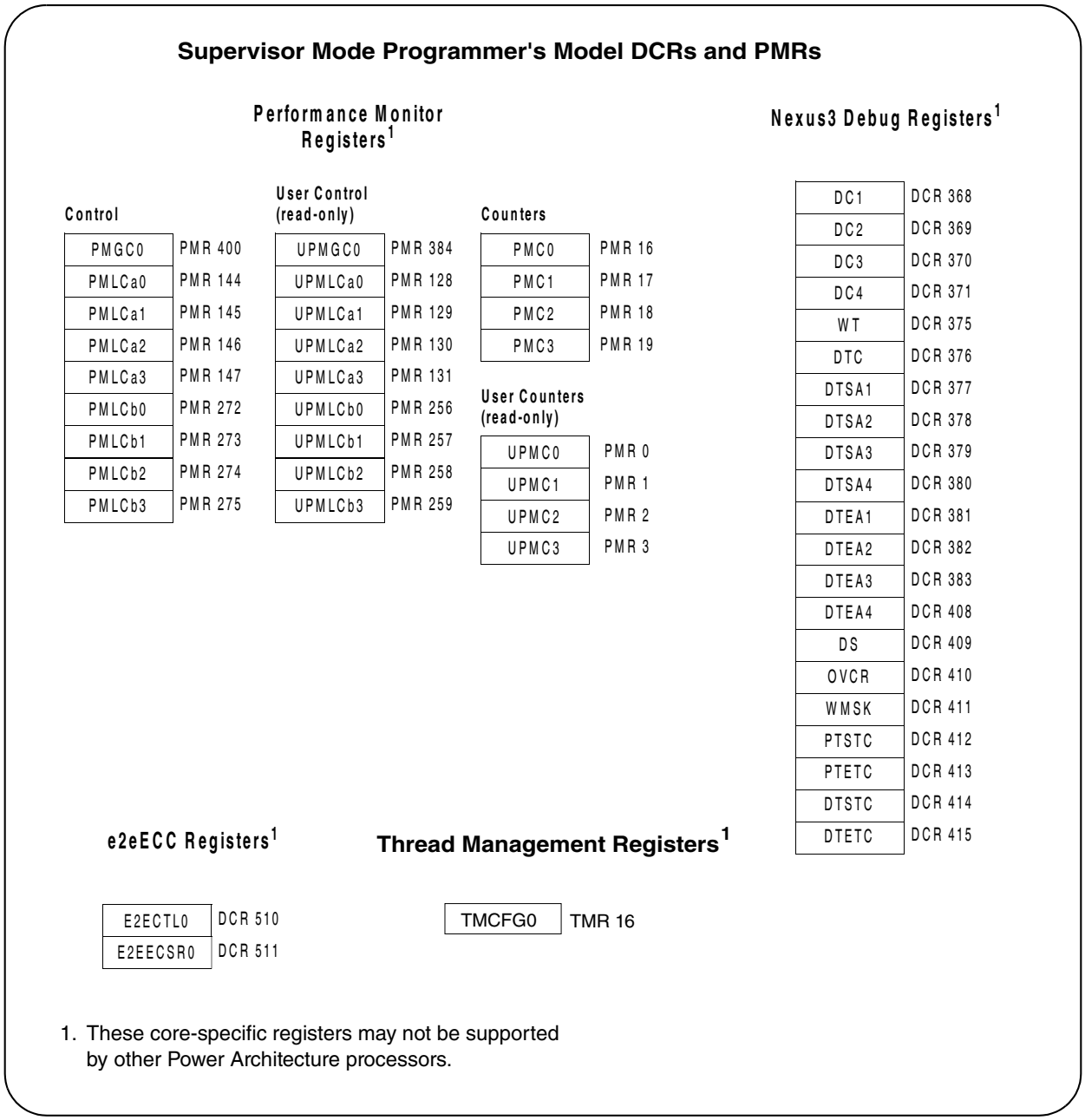


Figure 62-2. e200z210n3 Supervisor Mode Programmer's Model DCRs and PMRs

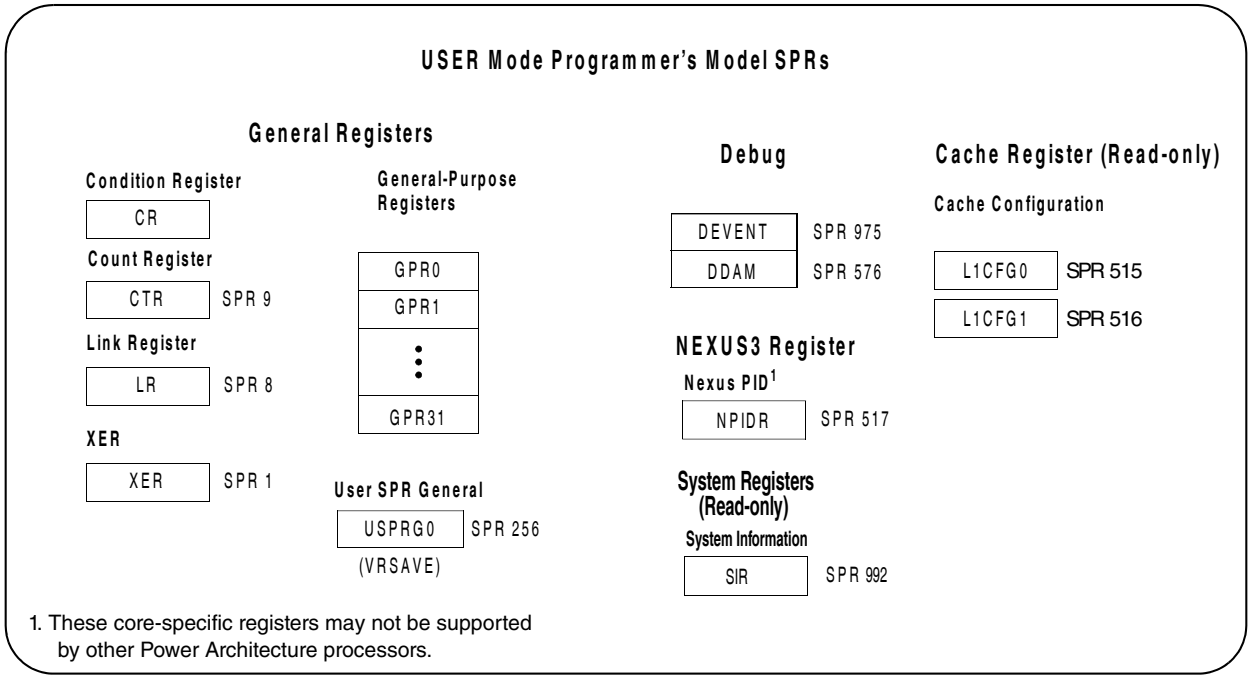


Figure 62-3. e200z210n3 User Mode Programmer's Model SPRs

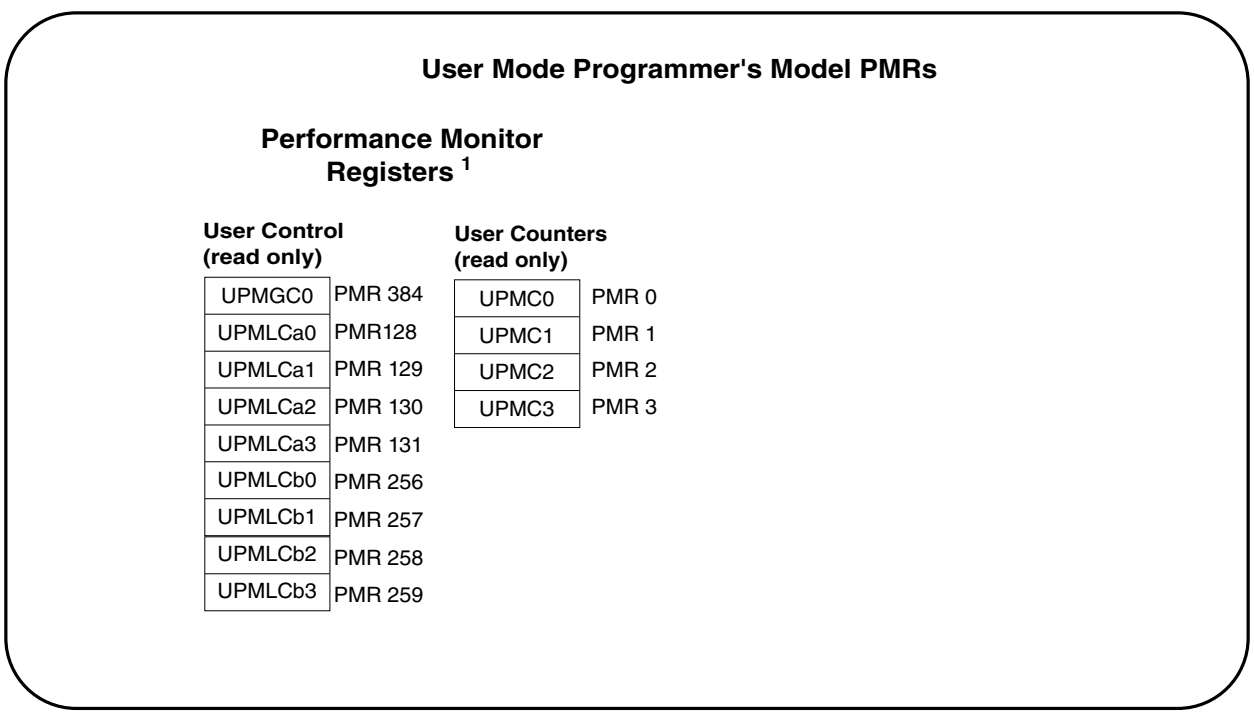


Figure 62-4. e200z210n3 User Mode Programmer's Model PMRs

62.2.1 Branch Unit Control and Status Register (BUCSR)

The BUCSR register is used for general control and status of the branch target buffer (BTB). The BTB is detailed in . The BUCSR is shown in the following figure.



SPR — 1013; Read/Write; Reset — 0x0

Figure 62-5. Branch Unit Control and Status Register (BUCSR)

The BUCSR fields are defined in the following table.

Table 62-1. Branch Unit Control and Status Register (BUCSR)

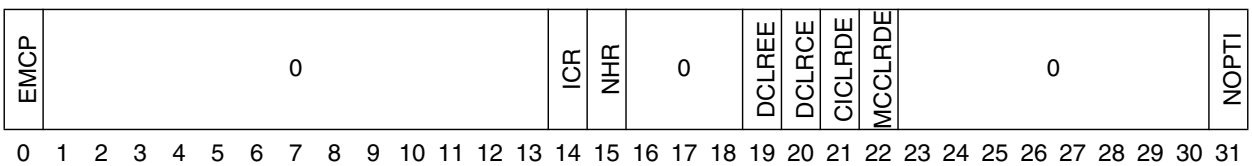
Bits	Name	Description
0:21 [32:53]	-	Reserved
22 [54]	BBFI	Branch target buffer flash invalidate. When written to a '1', BBFI flash clears the valid bit of all entries in the branch target buffer; clearing occurs regardless of the value of the enable bit (BPEN). Note: BBFI is always read as 0.
23:25 [55:57]	-	Reserved
26:27 [58:59]	BALLOC	Branch Target Buffer Allocation Control. This field controls BTB allocation for branch acceleration when BPEN = 1. Note that BTB hits are not affected by the settings of this field. 00 Branch Target Buffer allocation for all branches is enabled. 01 Branch Target Buffer allocation is disabled for backward branches. 10 Branch Target Buffer allocation is disabled for forward branches. 11 Branch Target Buffer allocation is disabled for both branch directions.
28 [60]	-	Reserved
29:30 [61:62]	BPRED	Branch Prediction Control (Static). This field controls operation of static prediction mechanism on a BTB miss. When a branch is predicted as taken, fetching of the predicted target location will be performed for branch acceleration. BPRED operates independently of BPEN, and with a BPEN setting of 0, will be used to perform static prediction of all unresolved branches. 00 Branch predicted taken on BTB miss for all branches. 01 Branch predicted taken on BTB miss only for forward branches. 10 Branch predicted taken on BTB miss only for backward branches. 11 Branch predicted not taken on BTB miss for both branch directions. **NOTE:** BTB hits are not affected by the settings of this field. Note also that for certain applications, setting BPRED to a non-default value may result in improved performance.
31 [63]	BPEN	Branch target buffer prediction enable. When the BPEN bit is cleared, no hits will be generated from the BTB, and no new entries will be allocated. Entries are not automatically invalidated when BPEN is cleared; the BBFI bit controls entry invalidation. BPEN operates independently of BPRED, and will be used even with a BPRED setting of 00.

Table 62-1. Branch Unit Control and Status Register (BUCSR)

Bits	Name	Description
		0 Branch target buffer prediction disabled 1 Branch target buffer prediction enabled (enables BTB to predict branches)

62.2.2 Hardware Implementation Dependent Register 0 (HID0)

The HID0 register is a core implementation dependent register used for various configuration and control functions. It is shown in the following figure.



SPR - 1008; Read/Write; Reset - 0x0

Figure 62-6. Hardware Implementation Dependent Register 0 (HID0)

The HID0 fields are defined in the following table.

Table 62-2. Hardware Implementation-Dependent Register 0

Bits	Name	Description
0 [32]	EMCP	Enable machine check pin (p_mcp_b) 0 p_mcp_b pin is disabled. 1 p_mcp_b pin is enabled. Asserting p_mcp_b causes a machine check interrupt to be reported.
1:13 [33:45]	-	Reserved
14 [46]	ICR	Interrupt Inputs Clear Reservation 0 External Input, Critical Input, and Non-Maskable Interrupts do not affect reservation status 1 External Input, Critical Input, and Non-Maskable Interrupts clear an outstanding reservation
15 [47]	NHR	Not hardware reset 0 indicates to a reset exception handler that a reset occurred if software had previously set this bit 1 indicates to a reset exception handler that no reset occurred if software had previously set this bit Provided for software use. Set anytime by software, cleared by reset.
16:18 [48:50]	-	Reserved
19 [51]	DCLREE	Debug Interrupt Clears MSR_{EE} 0 MSR_{EE} unaffected by Debug Interrupt 1 MSR_{EE} cleared by Debug Interrupt

Table continues on the next page...

Table 62-2. Hardware Implementation-Dependent Register 0 (continued)

Bits	Name	Description
		This bit controls whether Debug interrupts force External Input interrupts to be disabled, or whether they remain unaffected.
20 [52]	DCLRCE	Debug Interrupt Clears MSR_{CE} 0 MSR_{CE} unaffected by Debug Interrupt 1 MSR_{CE} cleared by Debug Interrupt This bit controls whether Debug interrupts force Critical interrupts to be disabled, or whether they remain unaffected.
21 [53]	CICLRDE	Critical Interrupt Clears MSR_{DE} 0 MSR_{DE} unaffected by Critical class interrupt 1 MSR_{DE} cleared by Critical class interrupt This bit controls whether Critical interrupts force Debug interrupts to be disabled, or whether they remain unaffected. Machine Check interrupts have a separate control bit. Note that if Critical Interrupt Debug events are enabled ($DBCR0_{CIRPT}$ set), and MSR_{DE} is set at the time of a Critical Input Critical interrupt, a debug event will be generated after the Critical Interrupt Handler has been fetched, and the Debug handler will be executed first. In this case, $DSRR0_{DE}$ will have been cleared, such that after returning from the debug handler, the Critical interrupt handler will not be run with MSR_{DE} enabled.
22 [54]	MCCLRDE	Machine Check Interrupt Clears MSR_{DE} 0 MSR_{DE} unaffected by Machine Check interrupt 1 MSR_{DE} cleared by Machine Check interrupt This bit controls whether Machine Check interrupts force Debug interrupts to be disabled, or whether they remain unaffected.
23:30 [58:62]	-	Reserved
31 [63]	NOPTI	No-op Touch Instructions 0 icbt, dcbt, dcbtst instructions operate normally 1 icbt, dcbt, dcbtst instructions are no-oped This bit only affects the icbt, dcbt, and dcbtst instructions.

62.2.3 Hardware Implementation Dependent Register 1 (HID1)

The HID1 register is used for bus configuration and system control. It is shown in the following figure.

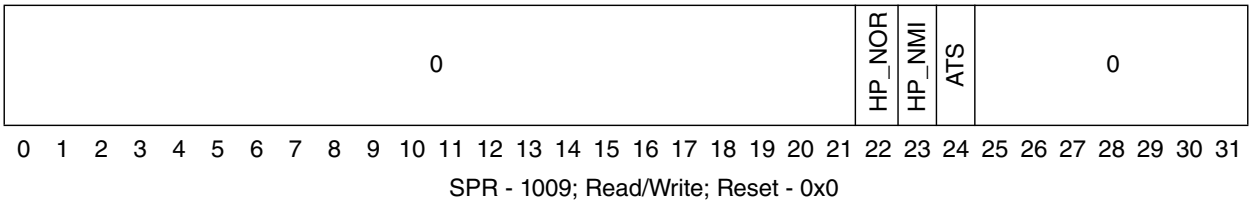


Figure 62-7. Hardware Implementation Dependent Register 1 (HID1)

The HID1 fields are defined in the following table.

Table 62-3. Hardware Implementation-Dependent Register 1

Bits	Name	Description
0:21 [32:53]	-	Reserved
22 [54]	HP_NOR	High priority elevation for normal and external interrupts 0 - High priority elevation for normal and external interrupts is disabled 1 - High priority elevation for normal and external interrupts is enabled The HP_NOR bitfield enables temporary elevation of the processor's crossbar priority. The relevant Master x field for the processor must also be enabled in the XBAR_CRSn HPEx bitfield. The platform logic elevates the processor's crossbar priority from the time of the IRQ assertion until the appropriate return instruction has been executed. For normal interrupts , the hardware supports the stacking of up to 4 interrupts. No hardware support of stacking for critical or NMI interrupts is provided. These bits may need external synchronization. **NOTE:** Must be used in conjunction with the XBAR_CRSn HPEx setting
23 [55]	HP_NMI	High priority elevation for NMI and critical interrupts 0 - High priority elevation for NMI and critical interrupts is disabled 1 - High priority elevation for NMI and critical interrupts is enabled The HP_NMI bitfield enables temporary elevation of the processor's crossbar priority. The relevant Master x field for the processor must also be enabled in the XBAR_CRSn HPEx bitfield. The platform logic elevates the processor's crossbar priority from the time of the IRQ assertion until the appropriate return instruction has been executed. For normal interrupts , the hardware supports the stacking of up to 4 interrupts. No hardware support of stacking for critical or NMI interrupts is provided These bits may need external synchronization. **NOTE:** Must be used in conjunction with the XBAR_CRSn HPEx setting
24 [56]	ATS	Atomic status (read-only) Indicates state of the reservation bit in the load/store unit.
25:31 [57:63]	-	Reserved

62.3 Single-issue operation

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

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

62.4 Instruction timing

Instruction timing in number of processor clock cycles for various instruction classes is shown in Table 62-4. Pipelined instructions are shown with cycles of total latency and throughput cycles. Divide and multiply instructions are not pipelined and block other instructions from executing during execution.

Load/store multiple instruction cycles are represented as a fixed number of cycles plus a variable number of cycles where *n* is the number of words accessed by the instruction. In addition, cycle times marked with & require variable number of additional cycles due to serialization.

Table 62-4. Instruction class cycle counts

Class of Instructions	Latency	Throughput	Special notes
integer: **add**, **sub**, **shift**, **rotate**, **logical**, **cntlzw**	1	1	—
integer: **compare**	1	1	—
Branch	2/1	2/1	Correct branch lookahead allows single-cycle execution. Worst-case mispredicted branch is 2 cycles.
multiply	2—4	2—4	Data-dependent timing
CR logical	1	1	—
loads (non-multiple)	1	1	—
load multiple	2 + *n*/2 (max)	2 + *n*/2 (max)	Actual timing depends on *n* and address alignment.
stores (non-multiple)	1	1	—

Table continues on the next page...

Table 62-4. Instruction class cycle counts (continued)

Class of Instructions	Latency	Throughput	Special notes
store multiple	2 + *n*/2 (max)	2 + *n*/2 (max)	Actual timing depends on *n* and address alignment.
mtmsr, **wrtee**, **wrteei**	3&	3&	—
mcrf	1	1	—
mfspr, **mtspr**	4&	4&	Applies to Debug SPRs, optional unit SPRs
mfspr, **mfmsr**	1	1	Applies to certain limited internal, non Debug SPRs
mfcr, **mtcr**	1	1	—
se_rfi, **se_rfci**, **se_rfdi**, **se_rfmci**	3	—	—
se_sc, **e_sc**	4	—	—
e_tw	4	—	Trap taken timing

Detailed timing for each instruction mnemonic along with serialization requirements is shown in Table 62-5 and Table 62-6.

Table 62-5. Instruction timing by mnemonic—16-bit instructions

Mnemonic	Latency	Serialization
se_add	1	—
se_addi	1	—
se_and[.]	1	—
se_andc	1	—
se_andi	1	—
se_bc	2/1	—
se_bclri	1	—
se_bctr	2	—
se_bctrl	2	—
se_bgeni	1	—
se_bl	2/1	—
se_blr	2	—
se_blrl	2	—
se_bmaski	1	—
se_b	2/1	—
se_bseti	1	—
se_btsti	1	—
se_cmp	1	—
se_cmph	1	—
se_cmphl	1	—
se_cmpi	1	—

Table continues on the next page...

Table 62-5. Instruction timing by mnemonic—16-bit instructions (continued)

Mnemonic	Latency	Serialization
se_cmpl	1	—
se_cmpli	1	—
se_dnh	1+	—
se_dni	1+	—
se_extsb	1	—
se_extsh	1	—
se_extzb	1	—
se_extzh	1	—
se_illegal	4	—
se_isync	6[1]	Refetch
se_lbz	1	—
se_lhz	1[2]	—
se_li	1	—
se_lwz	1[2]	—
se_mfar	1	—
se_mfctr	1	—
se_mflr	1	—
se_mr	1	—
se_mtar	1	—
se_mtctr	1	—
se_mtlr	1	—
se_mullw	2—4	—
se_neg	1	—
se_not	1	—
se_or	1	—
se_rfci	3	Refetch
se_rfdi	3	Refetch
se_rfi	3	Refetch
se_rfmci	3	Refetch
se_sc	4	Refetch
se_slw	1	—
se_slwi	1	—
se_sraw	1	—
se_srawi	1	—
se_srw	1	—
se_srwi	1	—
se_stb	1	—
se_sth	1[2]	—
se_stw	1[2]	—

Table continues on the next page...

Table 62-5. Instruction timing by mnemonic—16-bit instructions (continued)

Mnemonic	Latency	Serialization
se_sub	1	—
se_subf	1	—
se_subi[.]	1	—

1. Plus additional synchronization time.
2. Aligned

Table 62-6. Instruction timing by mnemonic—32-bit instructions

Mnemonic	Latency	Serialization
add[.]	1	—
e_add16i	1	—
e_add2i.	1	—
e_add2is	1	—
addc[.]	1	—
addco[.]	1	—
adde[.]	1	—
addeo[.]	1	—
e_addi[.]	1	—
e_addic[.]	1	—
addme[.]	1	—
addmeo[.]	1	—
addo[.]	1	—
addze[.]	1	—
addzeo[.]	1	—
and[.]	1	—
e_and2i.	1	—
e_and2is.	1	—
andc[.]	1	—
e_andi[.]	1	—
e_b	2/1	—
e_bc	2/1	—
e_bcl	2/1	—
e_bl	2/1	—
cmp	1	—
e_cmp16i	1	—
e_cmph	1	—
e_cmph16i	1	—
e_cmphl	1	—
e_cmphl16i	1	—
e_cmpi	1	—

Table continues on the next page...

Table 62-6. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
cmpl	1	—
e_cmpl16i	1	—
e_cmpli	1	—
cntlzw[.]	1	—
e_crand	1	—
e_crandc	1	—
e_creqv	1	—
e_crnand	1	—
e_crnor	1	—
e_cror	1	—
e_crorc	1	—
e_crxor	1	—
dcba	1	—
dcbf	1	—
dcbi	1	—
dcbst	1	—
dcbt	1	—
dcbtst	1	—
dcbz	— (alignment error)	—
divw[.]	5–34[1]	—
divwo[.]	5–34[1]	—
divwu[.]	5–34[1]	—
divwuo[.]	5–34[1]	—
e_dnh	1+	—
e_dni	1+	—
eqv[.]	1	—
extsb[.]	1	—
extsh[.]	1	—
icbi	1	—
icbt	1	—
isel	1	—
lbarx	1	—
e_lbz	1	—
e_lbzu	1	—
lbzux	1	—
lbzx	1	—
e_lha	1[2]	—
lharx	1	—
e_lhau	1[2]	—

Table continues on the next page...

Table 62-6. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
lhaux	1[2]	—
lhax	1[2]	—
lhbrx	1[2]	—
e_lhz	1[2]	—
e_lhzu	1[2]	—
lhzux	1[2]	—
lhzx	1[2]	—
e_li	1	—
e_lis	1	—
e_lmw	2 + (*n*/2)	—
lwarx	1	—
lwbrx	1[2]	—
e_lwz	1[2]	—
e_lwzu	1[2]	—
lwzux	1[2]	—
lwzx	1[2]	—
mbar	1[3]	Pseudo-dispatch
e_mcrf	1	—
mcrxr	1	Completion
mfcr	1	—
mfdcr	4[3]	Completion
mfmsr	1	—
mfspr (except DEBUG, CACHE, LMEM < MPU)	1	None
mfspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mpure	4[3]	Completion
mpusync	1	Completion
mpuwe	4[3]	Completion
msync	1[3]	Completion
mtcrf	2	—
mtmsr	3[3]	Completion
mtspr (except DEBUG, **msr**, hid0/1)	1	None
mtspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mulhw[.]	2—4	—
mulhwu[.]	2—4	—
e_mull2i	2—4	—
e_mulli	2—4	—
mullw[.]	2—4	—

Table continues on the next page...

Table 62-6. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
mullwo[.]	2—4	—
nand[.]	1	—
neg[.]	1	—
nego[.]	1	—
nor[.]	1	—
or[.]	1	—
e_or2i	1	—
e_or2is	1	—
orc[.]	1	—
e_ori[.]	1	—
e_rlw[.]	1	—
e_rlwi[.]	1	—
e_rlwimi	1	—
e_rlwinm	1	—
e_sc	4	Refetch
slw[.]	1	—
e_slwi[.]	1	—
sraw[.]	1	—
srawi[.]	1	—
srw[.]	1	—
e_srwi[.]	1	—
e_stb	1	—
stbcx.	1	—
e_stbu	1	—
stbux	1	—
stbx	1	—
e_sth	1[2]	—
sthbrx	1[2]	—
sthcx.	1	—
e_sthu	1[2]	—
sthux	1[2]	—
sthx	1[2]	—
e_stmw	2 + (*n*/2)	—
e_stw	1[2]	—
stwbrx	1[2]	—
stwcx.	1	—
e_stwu	1[2]	—
stwux	1[2]	—
stwx	1[2]	—

Table continues on the next page...

Table 62-6. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
subf[.]	1	—
subfc[.]	1	—
subfco[.]	1	—
subfe[.]	1	—
subfeo[.]	1	—
e_subfic[.]	1	—
subfme[.]	1	—
subfmeo[.]	1	—
subfo[.]	1	—
subfze[.]	1	—
subfzeo[.]	1	—
tw	4	—
wait	—	Completion
wrtee	3	Completion
wrteei	3	Completion
xor[.]	1	—
e_xori[.]	1	—

1. With early-out capability, timing is data-dependent.
2. Aligned.
3. Plus additional synchronization time.

62.5 Exceptions

Interrupts implemented in e200z210n3 and the exception conditions that cause them are listed in the following table.

Table 62-7. Exceptions and Conditions

Interrupt type	Interrupt vector offset value	Causing conditions
System reset	none, vector to [p_rstbase[0:29]] || 2'b00	Reset
Critical Input	0x00[1]	**p_critint_b** is asserted and MSR_{CE} = 1.
Machine check	0x10	1. **p_mcp_b** transitions from negated to asserted 2. ISI or Bus Error on first instruction fetch for an exception handler 3. External bus error 4. Stack limit check failure
Machine check (NMI)	0x10	Non-Maskable Interrupt

Table continues on the next page...

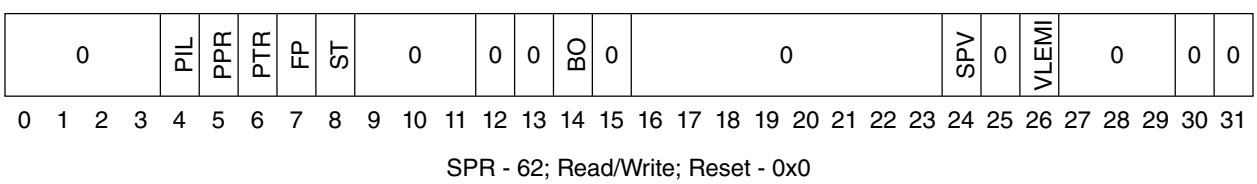
Table 62-7. Exceptions and Conditions (continued)

Interrupt type	Interrupt vector offset value	Causing conditions
Data Storage	0x20	Access control
Instruction Storage	0x30	Access control
External Input	0x40[2]	Interrupt Controller interrupt and MSR_{EE} = 1
Alignment	0x50	1. **lmw**, **stmw** not word aligned 2. **lwarx** or **stwcx.** not word aligned, **lharx** or **sthcx.** not halfword aligned 3. **dcbz**
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 0
System call	0x80	Execution of the System Call (**se_sc**) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/$PMGC0_{UDI}$ = 1
EFPU Data Exception	0xA0	Reserved for future processor use
EFPU Round Exception	0xB0	Reserved for future processor use
TBD	0xC0–0xF0	Reserved for future processor use

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

62.5.1 Exception Syndrome Register (ESR)

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



SPR - 62; Read/Write; Reset - 0x0

Figure 62-8. Exception Syndrome Register (ESR)

The ESR bits are defined in the following table.

Table 62-8. ESR field descriptions

Bit(s)	Name	Description	Associated interrupt type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception	Program

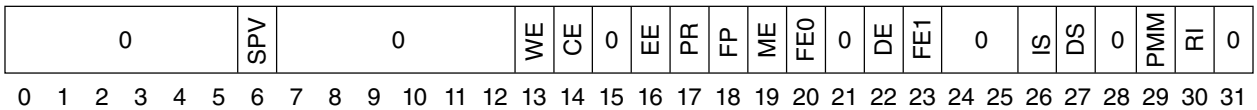
Table continues on the next page...

Table 62-8. ESR field descriptions (continued)

Bit(s)	Name	Description	Associated interrupt type
		For e200z210n3, PIL is used for both illegal and unimplemented instructions.	
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage
9:11	—	Reserved	—
12	—	Reserved	—
13	—	Reserved	—
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	—	Reserved	—
16:23	—	Reserved	—
24	SPV	EFPU APU Operation (Not used by Core (e200z210n3). Implemented but not set by hardware.)	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage Alignment Program System Call
27:29	—	Reserved	—
30	—	Reserved	—
31	—	Reserved	—

62.5.2 Machine State Register (MSR)

The Machine State Register defines the state of the processor. The e200z210n3 MSR is shown in the following figure.



Read/Write; Reset - 0x0

Figure 62-9. Machine State Register (MSR)

The MSR bits are defined in the following table.

Table 62-9. MSR field descriptions

Bit(s)	Name	Description
0:5	—	Reserved
6	SPV	SP/Embedded FP/Vector available 0 SP/Embedded FP Double/Vector unit is unavailable. The processor cannot execute SP/Embedded FP Double or Vector Unit instructions. 1 SP/Embedded FP Double/Vector unit is available. The processor can execute SP/Embedded FP Double or Vector Unit instructions. **NOTE:** The e200z210n3 processor does not support these units in hardware. An Illegal Instruction exception is generated for attempted execution of these instructions regardless of the setting of SPV. SPV is ignored, but cleared on exceptions.
7:12	—	Reserved
13	WE	Wait State (Power management) enable. **NOTE:** this bit is implementation dependent and is being phased out. It will be removed in a future release. Currently it is implemented as a writeable bit, and is ignored, but cleared on exceptions.
14	CE	Critical Interrupt Enable 0 Critical Input interrupts are disabled. 1 Critical Input interrupts are enabled.
15	—	Reserved
16	EE	External Interrupt Enable 0 External Input interrupts are disabled. 1 External Input interrupts are enabled.
17	PR	Problem State 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (e.g. GPRs, SPRs, MSR, etc.). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
18	FP	Floating-Point Available 0 Floating point unit is unavailable. The processor cannot execute floating-point instructions, including floating-point loads, stores, and moves. 1 Floating Point unit is available. The processor can execute floating-point instructions. **NOTE:** The e200z210n3 processor does not support the PowerISA 2.06 floating point unit in hardware, and an Illegal Instruction exception is generated for attempted execution of PowerISA 2.06 floating point instructions regardless of the setting of FP. FP is ignored, but cleared on exceptions.

Table continues on the next page...

Table 62-9. MSR field descriptions (continued)

Bit(s)	Name	Description
19	ME	Machine Check Enable 0 Asynchronous Machine Check interrupts are disabled. 1 Asynchronous Machine Check interrupts are enabled.
20	FE0	Floating-point exception mode 0 (not used) **NOTE:** The e200z210n3 processor does not support the PowerISA 2.06 floating point unit in hardware, thus FE0 is ignored, but cleared on exceptions.
21	—	Reserved
22	DE	Debug Interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23	FE1	Floating-point exception mode 1 (not used) **NOTE:** The e200z210n3 processor does not support the PowerISA 2.06 floating point unit in hardware, thus FE1 is ignored, but cleared on exceptions.
24:25	—	Reserved
26	IS	Instruction Address Space 0 The processor directs all instruction fetches to address space 0. 1 The processor directs all instruction fetches to address space 1. **NOTE:** The e200z210n3 processor does not support Address Spaces, thus the IS bit is ignored, but cleared on exceptions.
27	DS	Data Address Space 0 The processor directs all data storage accesses to address space 0. 1 The processor directs all data storage accesses to address space 1. **NOTE:** The e200z210n3 processor does not support Address Spaces, thus the DS bit is ignored, but cleared on exceptions.
28	—	Reserved
29	PMM	PMM Performance monitor mark bit System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR_{PR} and MSR_{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLCa*n* Performance Monitor registers, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken.
31	—	Reserved

62.5.3 Machine Check Syndrome Register (MCSR)

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

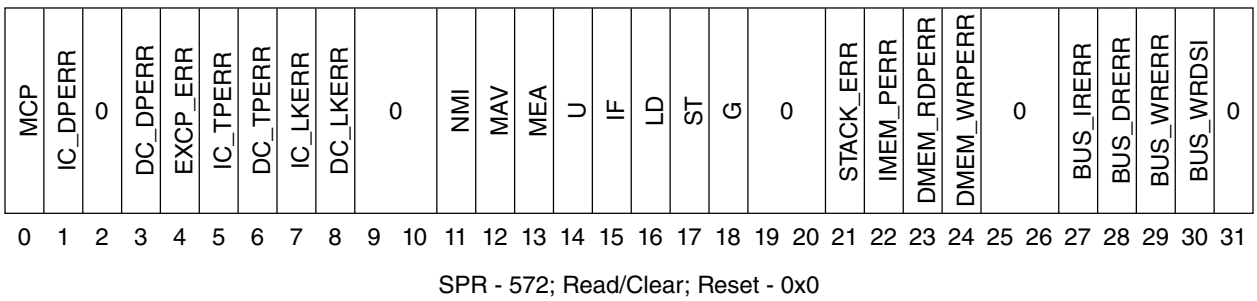


Figure 62-10. Machine Check Syndrome Register (MCSR)

The following table describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as "write 1 to clear." Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR_{ME} to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once $MSR_{ME} = 1$.

Table 62-10. Machine Check Syndrome Register (MCSR) field descriptions

Bit	Name	Description	Exception type[1]	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR[2]	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved	—	—
3	DC_DPERR[3]	Data Cache data array parity error	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR[2]	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR[3]	Data Cache Tag parity error	Async Mchk	Maybe
7	IC_LKERR[2]	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the I-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
8	DC_LKERR[3]	Data Cache Lock error	Async Mchk	—

Table continues on the next page...

Table 62-10. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type[1]	Recoverable
		Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the D-Cache for certain situations. May also be set on locked line refill error.		
9:10	—	Reserved	—	—
11	NMI	NMI input pin	NMI	—
12	MAV	MCAR Address Valid Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition	Status	—
13	MEA	MCAR holds Effective Address If MAV = 1, MEA = 1 indicates that the MCAR contains an effective address and MEA = 0 indicates that the MCAR contains a physical address[4]	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to a stack limit check error, or an external bus error.	Error Report	Precise
17	ST	Store type instruction Error Report An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to a stack limit check error, or on certain external bus errors.	Error Report	Precise
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus.	Error Report	Precise
19:20	—	Reserved	—	—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22	IMEM_PERR[5]	Instruction Mem (IMEM) Parity Error	Async Mchk	Precise

Table continues on the next page...

Table 62-10. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type[1]	Recoverable
		Indicates an uncorrectable error in the IMEM on a CPU port access.		
23	DMEM_RDPERR6	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise
24	DMEM_WRPERR6	Data Mem (DMEM) Write Parity Error Indicates an uncorrectable error in the DMEM on a CPU port write access.	Async Mchk	Maybe
25:26	—	Reserved	—	—
27	BUS_IRERR[2]	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved	—	—

1. The Exception Type indicates the exception type associated with a given syndrome bit.
   - "Error Report" indicates that this bit is only set for error report exceptions which cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR_{ME}. These are synchronous exceptions. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
   - "Status" indicates that this bit provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
   - "NMI" indicates that this bit is only set for the non-maskable interrupt type exception which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR_{ME}. This is an asynchronous exception. This bit will remain set until cleared by software writing a '1' to the bit position.
   - "Async Mchk" indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any "Async Mchk" bit is set in the MCSR, a machine check interrupt will occur if MSR_{ME} = 1. If MSR_{ME} = 0, the machine check exception will remain pending. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
2. Will not be set by e200z210n3 since no instruction cache is present.
3. Will not be set by e200z210n3 since no data cache is present.
4. Note that since no address translation mechanism is present on e200z210n3, the MEA value is always set to '0'.
5. Will not be set by e200z210n3 since no local instruction memory is present.
6. Will not be set by e200z210n3 since no local data memory is present.

62.5.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the IVPR to form an instruction address from which execution is to begin. The format of IVPR is shown in the following figure.

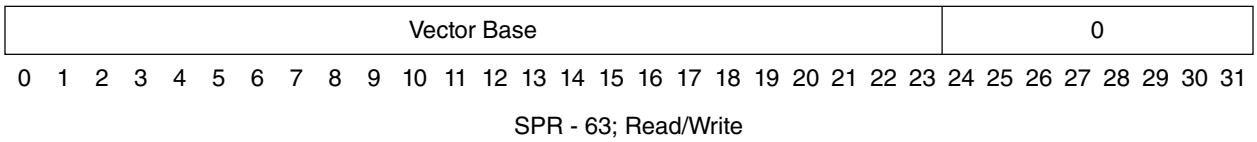


Figure 62-11. Interrupt Vector Prefix Register (IVPR)

The IVPR fields are defined in the following table.

Table 62-11. IVPR field descriptions

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256-byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically OR'ed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

62.5.5 Interrupt Definitions

62.5.5.1 Critical Input Interrupt (offset 0x00)

A Critical Input exception is signaled to the processor by the assertion of the critical interrupt pin. If the exception is enabled by MSR_{CE}, the Critical Input interrupt is taken.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

The following table lists register settings when a Critical Input interrupt is taken.

Table 62-12. Critical Input Interrupt—register settings

Register	Setting description					
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
CSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	—/0[1]	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x00 (autovectored)					
	$IVPR_{0:15}$ || (IVPR16:29 ⊕ **p_voffset[0:13]**) || 2'b00 (non-autovectored)					

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but it can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

62.5.5.2 Machine Check Interrupt (offset 0x10)

The EIS Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome Register (MCSR) to record the source(s) of machine checks, and a Machine Check Address Register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR_{DE} bit is not automatically cleared by a Machine Check exception, but it can be configured to be cleared or left unchanged via the HID0 register ($HID0_{MCCLRDE}$).

When a Machine Check interrupt is taken, registers are updated as shown in the following table.

Table 62-13. Machine Check Interrupt—register settings

Register	Setting description					
MCSRR0	On a best-effort basis, set to the address of some instruction that was executing or about to be executing when the machine check condition occurred.					
MCSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0

Table continues on the next page...

Table 62-13. Machine Check Interrupt—register settings (continued)

Register	Setting description					
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0/—[1]	PMM	0
	PR	0			RI	0
ESR	Unchanged					
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.					
Vector	$IVPR_{0:23}$ || 0x10					

1. Clearing of DE is optionally supported by control in HID0.

62.5.5.3 Data Storage Interrupt (offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

The following table lists register settings when a DSI is taken.

Table 62-14. Data Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Access:		[ST], VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	For Access Control exceptions, set to the effective address of the access that caused the violation.					
Vector	$IVPR_{0:23}$ || 0x20					

62.5.5.4 Instruction Storage Interrupt (offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

The following table lists register settings when an ISI is taken.

Table 62-15. Instruction Storage Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x30					

62.5.5.5 External Input Interrupt (offset 0x40)

An External Input exception is signaled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until the core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR_{EE}, it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR_{EE} is cleared when the exception occurs.

The following table lists register settings when an External Input interrupt is taken.

Table 62-16. External Input Interrupt—Register Settings

Register	Setting description					
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					

Table continues on the next page...

Table 62-16. External Input Interrupt—Register Settings (continued)

Register	Setting description
Vector	$IVPR_{0:23}$ || 0x40 (autovectored) IVPR0:15 || (IVPR16:29 | **p_voffset[0:13]**) || 2'b00 (non-autovectored)

62.5.5.6 Alignment Interrupt (offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** is not word aligned.
- The operand of **lwarx** or **stwcx.** is not word aligned.
- The operand of **lharx** or **sthcx.** is not halfword aligned.
- Execution of a **dcbz** instruction is attempted.

The following table lists register settings when an alignment interrupt is taken.

Table 62-17. Alignment Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store/dcbz instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	[ST], VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Set to the effective address of a byte of the load or store access causing the violation.					
Vector	$IVPR_{0:23}$ || 0x50					

62.5.5.7 Program Interrupt (offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception

- Privileged Instruction exception
- Trap exception

The e200z210n3 will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- An instruction from the illegal instruction class
- **mtspr** and **mfspr** instructions with an undefined SPR specified
- **mtdcr** and **mfdcr** instructions with an undefined DCR specified

The e200z210n3 will invoke a Privileged Instruction program exception on attempted execution of the following instructions when $MSR_{PR} = 1$ (user mode):

- A privileged instruction
- **mtspr** and **mfspr** instructions that specify a SPRN value with $SPRN_5 = 1$ (even if the SPR is undefined)

The e200z210n3 will invoke a Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

The core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfdcrx**, **mtdcrx**, or on any *Power ISA 2.06* floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause an illegal instruction program exception.

The following table lists register settings when a Program interrupt is taken.

Table 62-18. Program Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Illegal, Unimplemented:		PIL, VLEMI. All other bits cleared.			
	Privileged:		PPR, VLEMI. All other bits cleared.			

Table continues on the next page...

Table 62-18. Program Interrupt—register settings (continued)

Register	Setting description	
	Trap:	PTR, VLEMI. All other bits cleared.
MCSR	Unchanged	
DEAR	Unchanged	
Vector	$IVPR_{0:23}$ || 0x60	

62.5.5.8 Performance Monitor Interrupt (offset 0x70)

A performance monitor interrupt that may be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMC*x* register overflow condition occurs with the following settings:

- PMLCax_{CE} = 11; that is, for the given counter the overflow condition is enabled.
- PMCx_{OV} = 11; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, $PMGC0_{PMIE}$ must be set.

Although an exception condition may occur with MSR_{EE} = 0, the interrupt cannot be taken until MSR_{EE} = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

The following table lists register settings when an performance monitor interrupt is taken.

Table 62-19. Performance Monitor Interrupt—Register Settings

Register	Setting description					
SRR0/ DSRR0[1]	Set to the effective address of the next instruction to be executed.					
SRR1/ DSRR1[1]	Set to the contents of the MSR at the time of the interrupt.					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2]	FE0	0	DS	0
	EE	0/—[3]	DE	—/0[4]	PMM	0
	PR	0			RI	—
ESR	Unchanged					

Table continues on the next page...

Table 62-19. Performance Monitor Interrupt—Register Settings (continued)

Register	Setting description
MCSR	Unchanged
DEAR	Unchanged
Vector	$IVPR_{0:23}$ || 0x70

1. DSRR0/1 are used if $PMGC0_{UDI}$ = 1
2. CE is cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLRCE}$ = 1
3. EE is not cleared if $PMGC0_{UDI}$ = 1 and $HID0_{DCLREE}$ = 0
4. DE is cleared if $PMGC0_{UDI}$ = 1

62.5.5.9 System Call Interrupt (offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

The following table lists register settings when a System Call interrupt is taken.

Table 62-20. System Call Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction *following* the system call instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x80					

62.5.5.10 Debug Interrupt (offset 0x90)

There are multiple sources that can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both $DBCR0_{IDM}$ = 1 (internal debug mode) and MSR_{DE} = 1).

The following table lists register settings when a Debug interrupt is taken.

Table 62-21. Debug Interrupt—register settings

Register	Setting description					
DSRR0[1]	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP.Set to the effective address of the next instruction to be executed *following* the excepting instruction for DAC (usually) and ICMP. For a UDE, IRPT, CIRPT, DCNT, or DEVT type exception, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
DSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2]	FE0	0	DS	0
	EE	—/0[2]	DE	0	PMM	0
	PR	0			RI	—
DBSR[3]	Unconditional Debug Event:		UDE			
	Instr. Complete Debug Event:		ICMP			
	Branch Taken Debug Event:		BRT			
	Interrupt Taken Debug Event:		IRPT			
	Critical Interrupt Taken Debug Event:		CIRPT			
	Trap Instruction Debug Event:		TRAP			
	Instruction Address Compare:		{IAC}			
	Data Address Compare:		{DACR, DACW}			
	Debug Notify Interrupt:		DNI			
	Return Debug Event:		RET			
	Critical Return Debug Event:		CRET			
	External Debug Event:		{DEVT1, DEVT2}			
	Performance Monitor Debug Event:		PMI			
	and optionally, an Imprecise Debug Event flag		{IDE}			
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0x90					

1. Assumes that the Debug interrupt is precise
2. Conditional based on control bits in HID0
3. Note that multiple DBSR bits may be set

62.5.5.11 Embedded Floating-point Data Interrupt (offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and an EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception. Note that e200z210n3 does not utilize this interrupt, since no embedded floating-point unit is present.

The following table lists register settings when an EFPU Floating-point Data interrupt is taken.

Table 62-22. Embedded Floating-point Data Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0xA0					

62.5.5.12 Embedded Floating-point Round Interrupt (offset 0xB0)

The Embedded Floating-point Round interrupt is taken when an EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled. Note that e200z210n3 does not utilize this interrupt, since no embedded floating-point unit is present.

The following table lists register settings when an EFPU Floating-point Round interrupt is taken.

Table 62-23. Embedded Floating-point Round Interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0

Table continues on the next page...

Table 62-23. Embedded Floating-point Round Interrupt—register settings (continued)

Register	Setting description					
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	$IVPR_{0:23}$ || 0xB0					

62.5.5.13 System Reset Interrupt

The System Reset exception is a non-maskable, asynchronous exception signaled to the processor through the assertion of system-defined signals.

A System Reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

The following table lists register settings when a System Reset interrupt is taken.

Table 62-24. System Reset Interrupt—register settings

Register	Setting description					
CSRR0	Undefined					
CSRR1	Undefined					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0	PMM	0
	PR	0			RI	0
ESR	Cleared					
DEAR	Undefined					
Vector	[**p_rstbase[0:29]**] || 2'b00					

62.6 End-to-End ECC Support

This section describes end-to-end error detection and correction capability (e2eECC) enhancements to address additional diagnostic capabilities for the next generation of embedded designs.

62.6.1 End-to-End ECC control and configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) accessed using the **mfdcr** and **mtdcr** instructions. These registers and the operation of various features are described in the following subsections.

62.6.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.



DCR - 510; Read/Write; Reset - 0x5 || p_e2e_rstcfg[0:3]; Privileged

Figure 62-12. e2eECC Control Register 0 (E2ECTL0)

Table 62-25. E2ECTL0 field descriptions

Bits	Name	Description
0:27	—	Reserved
28	IDAC	Instruction Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Instruction AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[0]** input.
29	IECCEN	Instruction ECC Enable Field 0 End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed.

Table continues on the next page...

Table 62-25. E2ECTL0 field descriptions (continued)

Bits	Name	Description
		1 End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[1]** input.
30	DDAC	Data Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Data AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[2]** input.
31	DECCEN	Data ECC Enable Field 0 End-to-End ECC is disabled for the Data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices. 1 End-to-End ECC is enabled for performing error detection and correction on the Data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[3]** input.

e2eECC is enabled by setting the [I,D]ECCEN bit of the E2ECTL0 DCR to a 1. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for nonzero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the initialization, software may enable e2eECC operation by writing the appropriate values to the $E2ECTL0_{[I,D]ECCEN}$ control bits.

The $E2ECTL0_{[I,D]DAC}$ control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

62.6.1.2 End-to-End ECC fault injection by software

Fault injection provides a way to test error recovery and portions of the error detection/correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address, or checkbit error on external bus write cycles, and read back the written data to cause an error to be detected.

The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the **p_hwchkbit[7:0]** outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs **p_hwchkbit[7:0]** in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to the external bus. For the case of an external read access that receives an ERROR response (HRESP=ERR), the RCHKBIT field is written with all zeros.

Note that Nexus 3 accesses do not cause error injection or the capture of read checkbits, regardless of the settings of the E2EECSR0 register.

The following figure shows the layout of E2EECSR0.

0	RCHKBIT	0	WRC	INVC	0	WCHKBIT/CHKINVT
0 1 2 3	4 5 6 7 8 9 10 11	12 13 14 15	16	17	18 19	20 21 22 23	24 25 26 27 28 29 30 31

DCR - 511; Read/Write; Reset - 0x0; Privileged

Figure 62-13. e2eECC Error Control/Status Register 0 (E2EECSR0)

Table 62-26. E2EECSR0 field descriptions

Bits	Name	Description
0:3	—	Reserved
4:11	RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to external memory via the Data BIU. This field corresponds bit-wise to **p_hrchkbit[7:0]**. for external reads. Software may use this information to determine the stored checkbits of

Table continues on the next page...

Table 62-26. E2EECSR0 field descriptions (continued)

Bits	Name	Description
		external memories by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked). If the CPU receives an ERROR response (HRESP=ERR) on an external read access, the RCHKBIT field is written with all zeros.
12:15	—	Reserved
16:17	WRC	Write Control 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. Software must clear this field before rewriting to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10 by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved Note: Checkbit substitution has priority over Error injection
18:19	INVC	Invert Control 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying **p_hwchkbit[7:0]** based on CHKINVT. Software must clear this field before rewriting to 01 in order to generate a subsequent error injection operation. 10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying **p_hwchkbit[7:0]** based on CHKINVT. This field must be cleared by software to discontinue further fault injection. 11 Reserved Note: Checkbit substitution has priority over Error injection
20:23	—	Reserved
24:31	WCHKBIT/ CHKINVT	Write Checkbits / Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. This field corresponds bit-wise to **p_hwchkbit[7:0]**. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out on **p_hwchkbit[7:0]** when error injection occurs.

Chapter 63
Z4 Core description

63.1 Overview of the e200z4204n3 core

The e200z4204n3 is a dual-issue 32-bit PowerISA 2.06 VLE compliant design with 32-bit general-purpose registers (GPRs). The e200z4204n3 core implements the VLE (variable-length encoding) ISA, providing improved code density. The VLE ISA is further documented in *PowerISA 2.06* , a separate document.

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

The e200z4204n3 processor integrates a pair of integer execution units, a branch control unit, instruction fetch unit, load/store unit, and a multi-ported register file capable of sustaining six read and three write operations per clock cycle. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in many cases.

The e200z4204n3 contains an 8 KB Instruction Cache and a 4 KB Data Cache, as well as a Nexus Class 3+ real-time debug module with support for program and data trace features as well as extensive trace controls.

63.2 Register model

This section describes the registers implemented in the e200z4204n3 core. The PowerISA 2.06 architecture defines register-to-register operations for all computational instructions.

The e200z4204n3 extends usage of the general purpose registers to support EFPU operations on the 32-bit GPR registers.

Figure 63-1 and Figure 63-2 show the complete e200z4204n3 register set, all of which are accessible in supervisor mode. Figure 63-3 and Figure 63-4 show the subset of registers accessible in user mode. The number to the right of the special-purpose registers (SPRs) is the decimal number used in the instruction syntax to access the register (for example, the integer exception register (XER) is SPR 1).

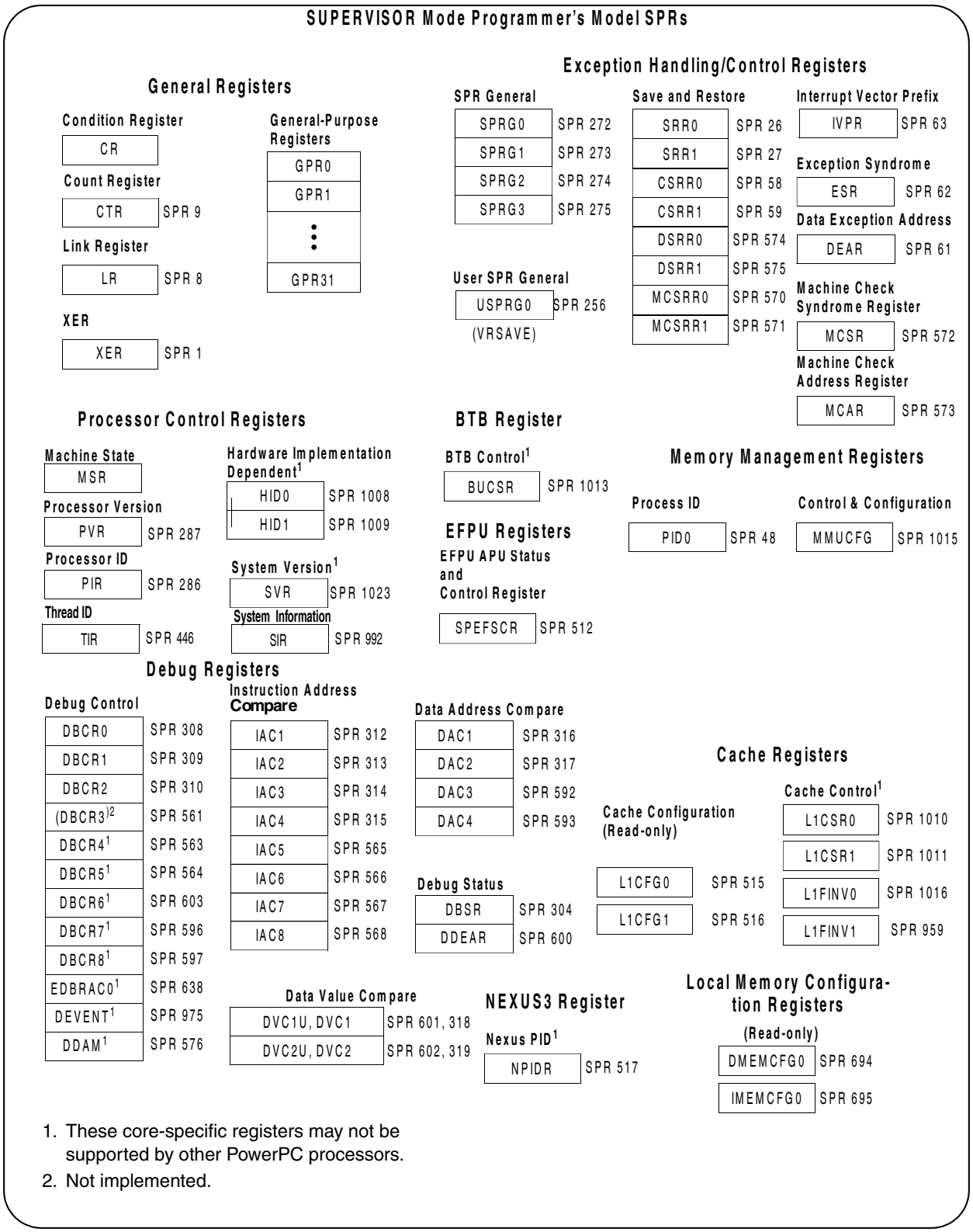


Figure 63-1. e200z4204n3 Supervisor Mode Programmer's Model SPRs

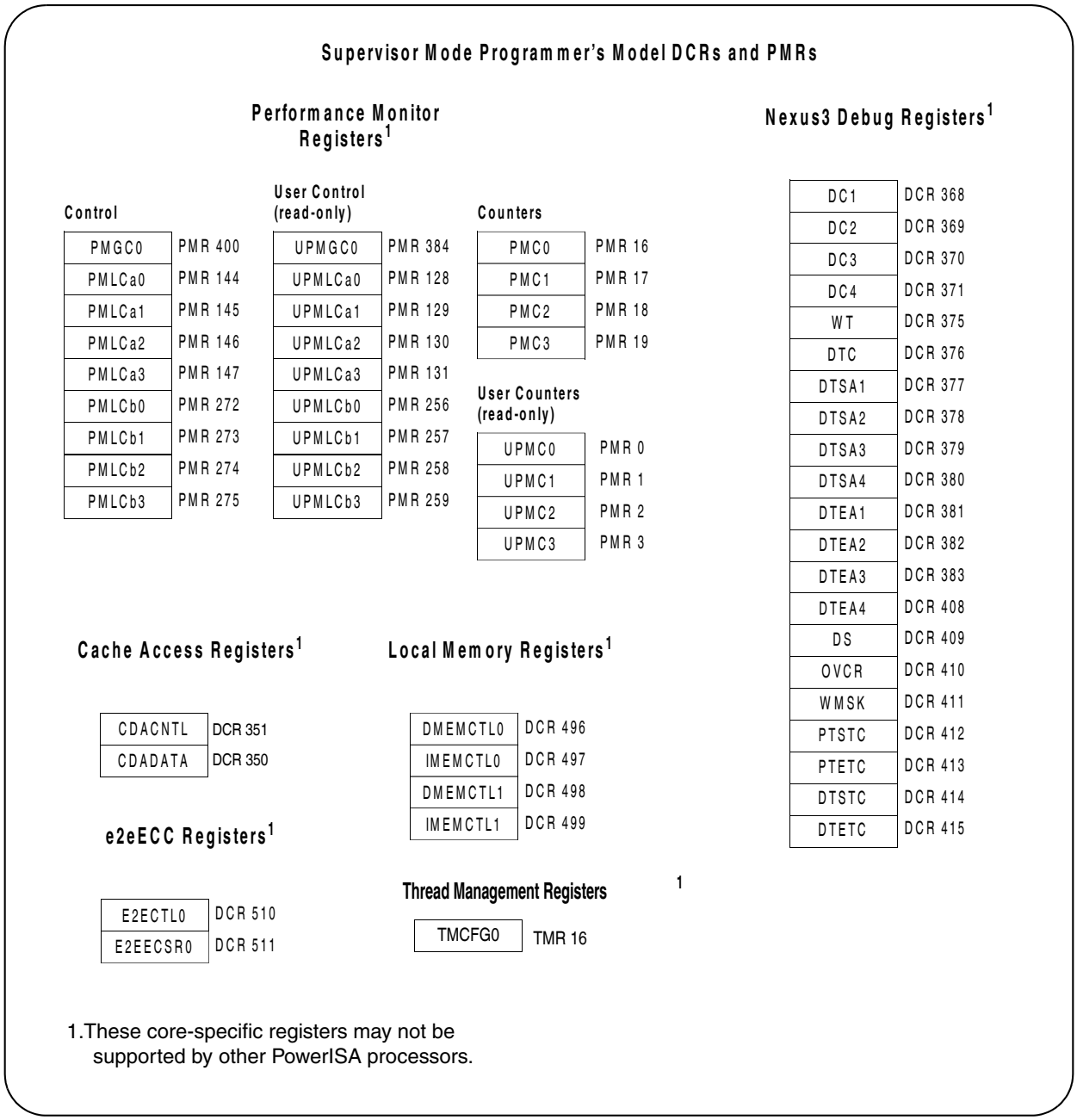


Figure 63-2. e200z4204n3 Supervisor Mode Programmer's Model DCRs and PMRs

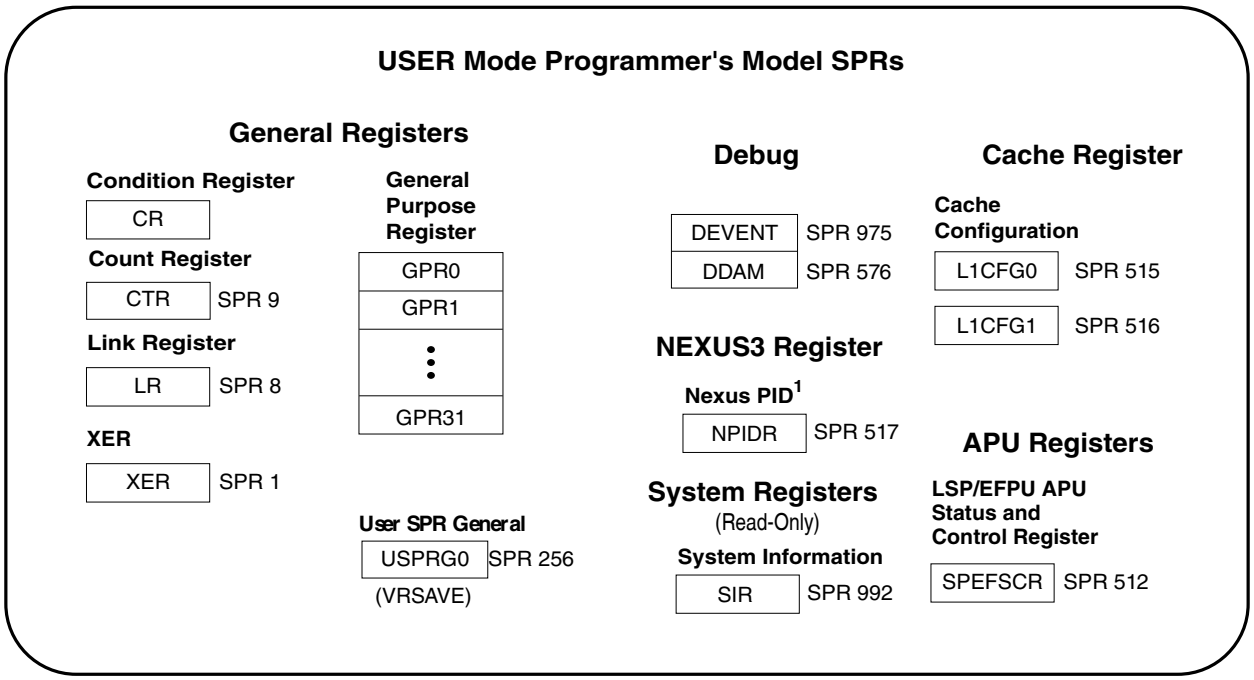


Figure 63-3. e200z4204n3 User Mode Programmer's Model SPRs

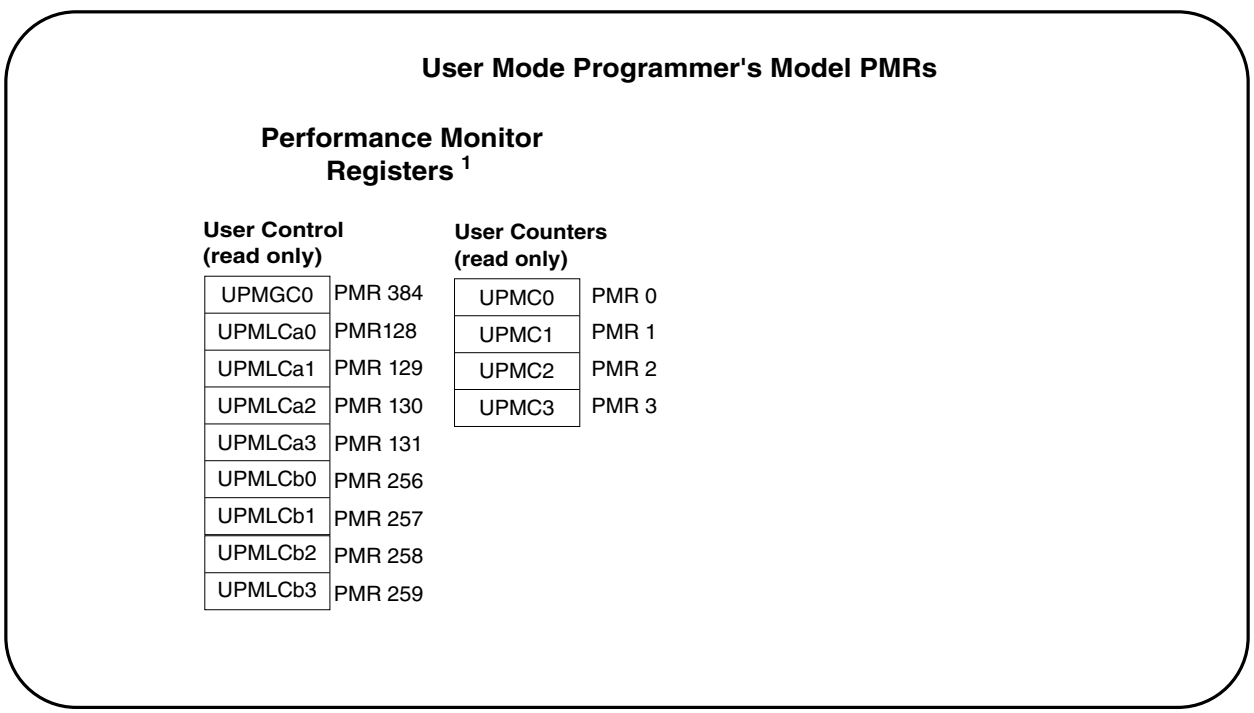
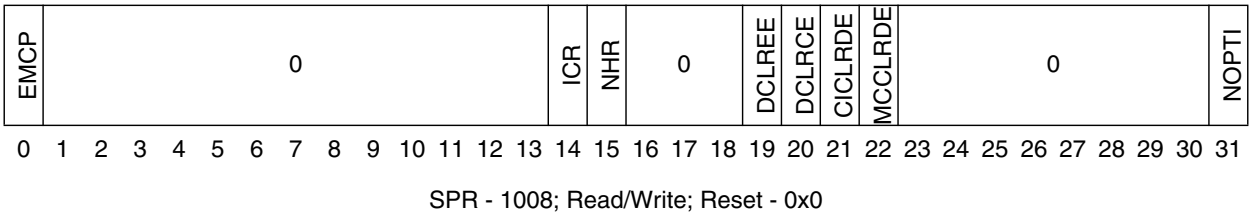


Figure 63-4. e200z4204n3 User Mode Programmer's Model PMRs

63.2.1 Hardware Implementation Dependent Register 0 (HID0)

The HID0 register is a core implementation dependent register used for various configuration and control functions. It is shown in the following figure.



SPR - 1008; Read/Write; Reset - 0x0

Figure 63-5. Hardware Implementation Dependent Register 0 (HID0)

The HID0 fields are defined in the following table.

Table 63-1. Hardware Implementation-Dependent Register 0

Bits	Name	Description
0 [32]	EMCP	Enable machine check pin (p_mcp_b) 0 p_mcp_b pin is disabled. 1 p_mcp_b pin is enabled. Asserting p_mcp_b causes a machine check interrupt to be reported.
1:13 [33:45]	-	Reserved
14 [46]	ICR	Interrupt Inputs Clear Reservation 0 External Input, Critical Input, and Non-Maskable Interrupts do not affect reservation status 1 External Input, Critical Input, and Non-Maskable Interrupts clear an outstanding reservation
15 [47]	NHR	Not hardware reset 0 indicates to a reset exception handler that a reset occurred if software had previously set this bit 1 indicates to a reset exception handler that no reset occurred if software had previously set this bit Provided for software use. Set anytime by software, cleared by reset.
16:18 [48:50]	-	Reserved
19 [51]	DCLREE	Debug Interrupt Clears MSR_{EE} 0 MSR_{EE} unaffected by Debug Interrupt 1 MSR_{EE} cleared by Debug Interrupt This bit controls whether Debug interrupts force External Input interrupts to be disabled, or whether they remain unaffected.
20 [52]	DCLRCE	Debug Interrupt Clears MSR_{CE} 0 MSR_{CE} unaffected by Debug Interrupt 1 MSR_{CE} cleared by Debug Interrupt This bit controls whether Debug interrupts force Critical interrupts to be disabled, or whether they remain unaffected.
21 [53]	CICLRDE	Critical Interrupt Clears MSR_{DE} 0 MSR_{DE} unaffected by Critical class interrupt 1 MSR_{DE} cleared by Critical class interrupt This bit controls whether Critical interrupts force Debug interrupts to be disabled, or whether they remain unaffected. Machine Check interrupts have a separate control bit.

Table continues on the next page...

Table 63-1. Hardware Implementation-Dependent Register 0 (continued)

Bits	Name	Description
		Note that if Critical Interrupt Debug events are enabled ($DBCR0_{CIRPT}$ set), and MSR_{DE} is set at the time of a Critical Input Critical interrupt, a debug event will be generated after the Critical Interrupt Handler has been fetched, and the Debug handler will be executed first. In this case, $DSRR0_{DE}$ will have been cleared, such that after returning from the debug handler, the Critical interrupt handler will not be run with MSR_{DE} enabled.
22 [54]	MCCLRDE	Machine Check Interrupt Clears MSR_{DE} 0 MSR_{DE} unaffected by Machine Check interrupt 1 MSR_{DE} cleared by Machine Check interrupt This bit controls whether Machine Check interrupts force Debug interrupts to be disabled, or whether they remain unaffected.
23:30 [58:62]	-	Reserved
31 [63]	NOPTI	No-op Touch Instructions 0 icbt, dcbt, dcbtst instructions operate normally 1 icbt, dcbt, dcbtst instructions are no-oped This bit only affects the icbt, dcbt, and dcbtst instructions.

63.2.2 Hardware Implementation Dependent Register 1 (HID1)

The HID1 register is used for bus configuration and system control. It is shown in the following figure.

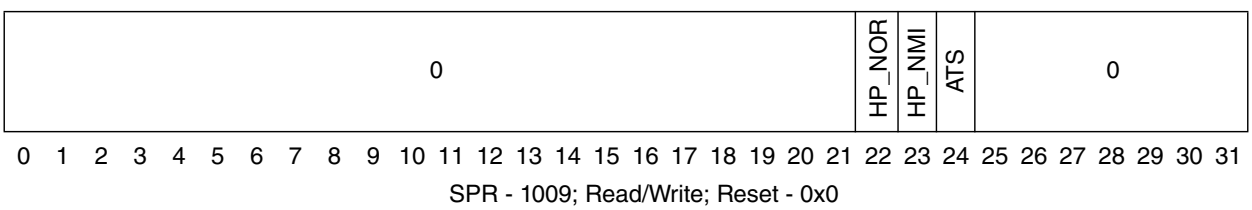


Figure 63-6. Hardware Implementation Dependent Register 1 (HID1)

The HID1 fields are defined in the following table.

Table 63-2. Hardware Implementation-Dependent Register 1

Bits	Name	Description
0:21 [32:53]	-	Reserved
22 [54]	HP_NOR	High priority elevation for normal and external interrupts 0 - High priority elevation for normal and external interrupts is disabled 1 - High priority elevation for normal and external interrupts is enabled

Table continues on the next page...

Table 63-2. Hardware Implementation-Dependent Register 1 (continued)

Bits	Name	Description
		The HP_NOR bitfield enables temporary elevation of the processor's crossbar priority. The relevant Master x field for the processor must also be enabled in the XBAR_CRSn HPEx bitfield. The platform logic elevates the processor's crossbar priority from the time of the IRQ assertion until the appropriate return instruction has been executed. For normal interrupts , the hardware supports the stacking of up to 4 interrupts. No hardware support of stacking for critical or NMI interrupts is provided. These bits may need external synchronization. **NOTE:** Must be used in conjunction with the XBAR_CRSn HPEx setting
23 [55]	HP_NMI	High priority elevation for NMI and critical interrupts 0 - High priority elevation for NMI and critical interrupts is disabled 1 - High priority elevation for NMI and critical interrupts is enabled The HP_NMI bitfield enables temporary elevation of the processor's crossbar priority. The relevant Master x field for the processor must also be enabled in the XBAR_CRSn HPEx bitfield. The platform logic elevates the processor's crossbar priority from the time of the IRQ assertion until the appropriate return instruction has been executed. For normal interrupts , the hardware supports the stacking of up to 4 interrupts. No hardware support of stacking for critical or NMI interrupts is provided These bits may need external synchronization. **NOTE:** Must be used in conjunction with the XBAR_CRSn HPEx setting
24 [56]	ATS	Atomic status (read-only) Indicates state of the reservation bit in the load/store unit.
25:31 [57:63]	-	Reserved

63.3 Instruction timing

Instruction timing in number of processor clock cycles for various instruction classes is shown in the following table. Pipelined instructions are shown with cycles of total latency and throughput cycles. Divide and multiply instructions are not pipelined and block other instructions from executing during execution. Load/store multiple instruction cycles are represented as a fixed number of cycles plus a variable number of cycles where *n* is the number of words accessed by the instruction. In addition, cycle times marked with & require variable number of additional cycles due to serialization.

Table 63-3. Instruction class cycle counts

Class of Instructions	Latency	Throughput	Special Notes
integer: **add, sub, shift, rotate, logical, cntlzw**	1	1	
integer: compare	1	1	

Table continues on the next page...

Table 63-3. Instruction class cycle counts (continued)

Class of Instructions	Latency	Throughput	Special Notes
Branch	3/2/1	3/2/1	Correct branch lookahead allows single cycle execution worst-case mispredicted branch is 3 cycles
multiply (saturating/non-saturating)	2	1	
divide	4-14	4-14	
CR logical	1	1	
loads (non-multiple)	2	1	
load multiple	2 + n/2 (max)	2 + n/2 (max)	Actual timing depends on n and address alignment
stores (non-multiple)	2	1	
store multiple	2 + n/2 (max)	2 + n/2 (max)	Actual timing depends on n and address alignment
mtmsr, wrtee, wrteei	3&	3&	
mcrf	1	1	
mfspr, mtspr	4&	4&	Applies to Debug SPRs, optional unit SPRS
mfspr, mfmsr	1	1	applies to certain limited internal, non Debug SPRs
mfcr, mtcr	1	1	
se_rfi, se_rfci, se_rfdi, se_rfmci	3	-	
se_sc, e_sc	4	-	
e_tw	4	-	Trap takem timing

Table 63-4. Instruction timing by mnemonic—16-bit instructions

Mnemonic	Latency	Serialization
se_add	1	—
se_addi	1	—
se_and[.]	1	—
se_andc	1	—
se_andi	1	—
se_bc	3/2/1	—
se_bclri	1	—
se_bctr	3/2	—
se_bctrl	3/2	—
se_bgeni	1	—
se_bl	3/2/1	—
se_blr	3/2	—
se_blrl	3/2	—
se_bmaski	1	—

Table continues on the next page...

Table 63-4. Instruction timing by mnemonic—16-bit instructions (continued)

Mnemonic	Latency	Serialization
se_b	3/2/1	—
se_bseti	1	—
se_btsti	1	—
se_cmp	1	—
se_cmph	1	—
se_cmphl	1	—
se_cmpi	1	—
se_cmpl	1	—
se_cmpli	1	—
se_dnh	1+	—
se_dni	1+	—
se_extsb	1	—
se_extsh	1	—
se_extzb	1	—
se_extzh	1	—
se_illegal	4	—
se_isync	6[1]	Refetch
se_lbz	2	—
se_lhz	2[2]	—
se_li	1	—
se_lwz	2[2]	—
se_mfar	1	—
se_mfctr	1	—
se_mflr	1	—
se_mr	1	—
se_mtar	1	—
se_mtctr	1	—
se_mtlr	1	—
se_mullw	2	—
se_neg	1	—
se_not	1	—
se_or	1	—
se_rfci	3	Refetch
se_rfdi	3	Refetch
se_rfi	3	Refetch
se_rfmci	3	Refetch
se_sc	4	Refetch
se_slw	1	—
se_slwi	1	—

Table continues on the next page...

Table 63-4. Instruction timing by mnemonic—16-bit instructions (continued)

Mnemonic	Latency	Serialization
se_sraw	1	—
se_srawi	1	—
se_srw	1	—
se_srwi	1	—
se_stb	2	—
se_sth	2[2]	—
se_stw	2[2]	—
se_sub	1	—
se_subf	1	—
se_subi[.]	1	—

1. Plus additional synchronization time.
2. Aligned

Table 63-5. Instruction timing by mnemonic—32-bit instructions

Mnemonic	Latency	Serialization
add[.]	1	—
e_add16i	1	—
e_add2i.	1	—
e_add2is	1	—
addc[.]	1	—
addco[.]	1	—
adde[.]	1	—
addeo[.]	1	—
e_addi[.]	1	—
e_addic[.]	1	—
addme[.]	1	—
addmeo[.]	1	—
addo[.]	1	—
addze[.]	1	—
addzeo[.]	1	—
and[.]	1	—
e_and2i.	1	—
e_and2is.	1	—
andc[.]	1	—
e_andi[.]	1	—
e_b	3/2/1	—
e_bc	3/2/1	—
e_bcl	3/2/1	—
e_bl	3/2/1	—

Table continues on the next page...

Table 63-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
cmp	1	—
e_cmp16i	1	—
e_cmph	1	—
e_cmph16i	1	—
e_cmphl	1	—
e_cmphl16i	1	—
e_cmpi	1	—
cmpl	1	—
e_cmpl16i	1	—
e_cmpli	1	—
cntlzw[.]	1	—
e_crand	1	—
e_crandc	1	—
e_creqv	1	—
e_crnand	1	—
e_crnor	1	—
e_cror	1	—
e_crorc	1	—
e_crxor	1	—
dcba	1	—
dcbf	1	—
dcbi	1	—
dcbst	1	—
dcbt	1	—
dcbtst	1	—
dcbz	— (alignment error)	—
divw[.]	4-14[1]	—
divwo[.]	4-14[1]	—
divwu[.]	4-14[1]	—
divwuo[.]	4-14[1]	—
e_dnh	1+	—
e_dni	1+	—
eqv[.]	1	—
extsb[.]	1	—
extsh[.]	1	—
icbi	1	—
icbt	1	—
isel	1	—
lbarx	2	—

Table continues on the next page...

Table 63-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
e_lbz	2	—
e_lbzu	2	—
lbzux	2	—
lbzx	2	—
e_lha	2[2]	—
lharx	2	—
e_lhau	2[2]	—
lhaux	2[2]	—
lhax	2[2]	—
lhbrx	2[2]	—
e_lhz	2[2]	—
e_lhzu	2[2]	—
lhzux	2[2]	—
lhzx	2[2]	—
e_li	1	—
e_lis	1	—
e_lmw	2+(n/2)	—
lwarx	2	—
lwbrx	2[2]	—
e_lwz	2[2]	—
e_lwzu	2[2]	—
lwzux	2[2]	—
lwzx	2[2]	—
mbar	1[3]	Pseudo-dispatch
e_mcrf	1	—
mcrxr	1	Completion
mfcr	1	—
mfdcr	4[3]	Completion
mfmsr	1	—
mfspr (except DEBUG, CACHE, LMEM < MPU)	1	None
mfspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mpure	4[3]	Completion
mpusync	1	Completion
mpuwe	4[3]	Completion
msync	1[3]	Completion
mtcrf	2	—
mtmsr	3[3]	Completion
mtspr (except DEBUG, **msr**, hid0/1)	1	None

Table continues on the next page...

Table 63-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
mtspr (DEBUG, CACHE, LMEM < MPU)	4[3]	Completion
mulhw[.]	2	—
mulhwu[.]	2	—
e_mull2i	2	—
e_mulli	2	—
mullw[.]	2	—
mullwo[.]	2	—
nand[.]	1	—
neg[.]	1	—
nego[.]	1	—
nor[.]	1	—
or[.]	1	—
e_or2i	1	—
e_or2is	1	—
orc[.]	1	—
e_ori[.]	1	—
e_rlw[.]	1	—
e_rlwi[.]	1	—
e_rlwimi	1	—
e_rlwinm	1	—
e_sc	4	Refetch
slw[.]	1	—
e_slwi[.]	1	—
sraw[.]	1	—
srawi[.]	1	—
srw[.]	1	—
e_srwi[.]	1	—
e_stb	2	—
stbcx.	2	—
e_stbu	2	—
stbux	2	—
stbx	2	—
e_sth	2[2]	—
sthbrx	2[2]	—
sthcx.	2	—
e_sthu	2[2]	—
sthux	2[2]	—
sthx	2[2]	—
e_stmw	2+(n/2)	—

Table continues on the next page...

Table 63-5. Instruction timing by mnemonic—32-bit instructions (continued)

Mnemonic	Latency	Serialization
e_stw	2[2]	—
stwbrx	2[2]	—
stwcx.	2	—
e_stwu	2[2]	—
stwux	2[2]	—
stwx	2[2]	—
subf[.]	1	—
subfc[.]	1	—
subfco[.]	1	—
subfe[.]	1	—
subfeo[.]	1	—
e_subfic[.]	1	—
subfme[.]	1	—
subfmeo[.]	1	—
subfo[.]	1	—
subfze[.]	1	—
subfzeo[.]	1	—
tw	4	—
wait	—	Completion
wrtee	3	Completion
wrteei	3	Completion
xor[.]	1	—
e_xori[.]	1	—

1. With early-out capability, timing is data-dependent.
2. Aligned.
3. Plus additional synchronization time.

63.4 Dual-issue operation

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

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

Two execution units are provided to allow dual issue of most instructions. Only a single load/store unit is provided. Only a single integer multiply and divide unit is provided, thus a pair of multiply or divide instructions cannot issue simultaneously. In addition, the divide unit is blocking.

The following table shows the concurrent instruction issue capabilities. Note that data dependencies between instructions will generally preclude dual-issue of those instructions.

Table 63-6. Concurrent instruction issue capabilities

Class of instruction	Branch	Load/Store	Scalar integer	Scalar float	Special
Branch	—	yes	yes	yes	—
Load/Store	yes	—	yes	yes	—
Scalar integer	yes	yes	yes[1]	yes	—
Scalar float	yes	yes	yes	—	—
Special	—	—	—	—	—

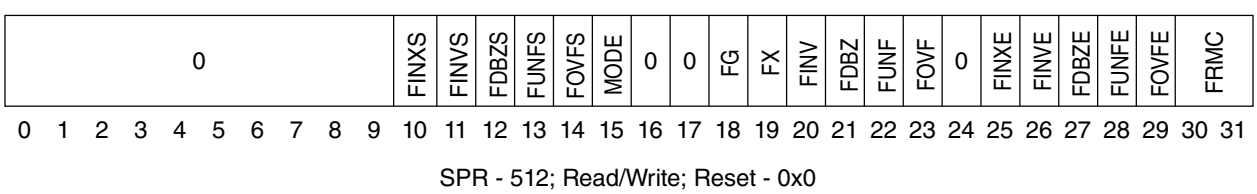
1. Excludes multiply or divide class instructions occurring in both issue slots.

63.5 Reservation instructions and cache interactions

If the CPU supports reservation instruction functionality, the CPU treats reservation instruction (**lbarx**, **lharx**, **lwarx**, **stbcx**, **sthcx**, and **stwcx**) accesses as though they were cache-inhibited, and forces a cache miss. A reservation access is always issued to the bus. This is done to allow external reservation logic to be built that properly signals a reservation failure. The bus access will be treated as a single-beat transfer.

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

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



SPR - 512; Read/Write; Reset - 0x0

Figure 63-7. EFPU Status and Control Register (SPEFSCR)

The SPEFSCR bits are defined in the following table.

Table 63-7. EFPU Status and Control Register field descriptions

Bits	Name	Description
0:9 (32:41)	—	Reserved
10 (42)	FINXS	Embedded Floating-point Inexact Sticky Flag The FINXS bit is set to 1 whenever the execution of a floating-point instruction delivers an inexact result and no Floating-point Data exception is taken, or if the result of a Floating-point instruction results in overflow (FOVF = 1), but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1), but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. The FINXS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
11 (43)	FINVS	Embedded Floating-point Invalid Operation Sticky Flag The FINVS bit is set to a 1 when a floating-point instruction sets the FINV bit to 1. The FINVS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
12 (44)	FDBZS	Embedded Floating-point Divide by Zero Sticky Flag The FDBZS bit is set to 1 when a floating-point divide instruction sets the FDBZ bit to 1. The FDBZS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
13 (45)	FUNFS	Embedded Floating-point Underflow Sticky Flag The FUNFS bit is set to 1 when a floating-point instruction sets the FUNF bit to 1. The FUNFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
14 (46)	FOVFS	Embedded Floating-point Overflow Sticky Flag The FOVFS bit is set to 1 when a floating-point instruction sets the FOVF bit to 1. The FOVFS bit remains set until it is cleared by a **mtspr** instruction specifying the SPEFSCR.
15 (47)	MODE	Embedded Floating-point Operating Mode This bit controls the operating mode of the EFPU. The e200z4204n3 supports only mode 0. Software should read the value of this bit after writing it to determine if the implementation supports the selected mode. Implementations will return the value written if the selected mode is a supported mode, otherwise the value read will indicate the hardware supported mode. 0 Default hardware results operating mode 1 IEEE754 hardware results operating mode (not supported by e200 core)
16 (48)	—	Reserved
17 (49)	—	Reserved
18 (50)	FG	Embedded Floating-point Guard bit FG is supplied for use by the Floating-point Round exception handler. FG is zeroed if a Floating-point Data Exception occurs.
19 (51)	FX	Embedded Floating-point Sticky bit FX is supplied for use by the Floating-point Round exception handler. FX is zeroed if a Floating-point Data Exception occurs.
20 (52)	FINV	Embedded Floating-point Invalid Operation / Input error

Table continues on the next page...

Table 63-7. EFPU Status and Control Register field descriptions (continued)

Bits	Name	Description
		In mode 0, the FINV bit is set to 1 if the A or B operand of a floating-point instruction is Infinity, NaN, or Denorm, or if the operation is a divide and the dividend and divisor are both 0. In mode 1, the FINV bit is set on an IEEE754 invalid operation (IEEE754-1985 sec7.1).
21 (53)	FDBZ	Embedded Floating-point Divide by Zero The FDBZ bit is set to 1 when a floating-point divide instruction executed with divisor of 0, and the l dividend is a finite non-zero number.
22 (54)	FUNF	Embedded Floating-point Underflow The FUNF bit is set to 1 when the execution of a floating-point instruction results in an underflow.
23 (55)	FOVF	Embedded Floating-point Overflow The FOVF bit is set to 1 when the execution of a floating-point instruction results in an overflow.
24 (56)	—	Reserved
25 (57)	FINXE	Embedded Floating-point Inexact Exception Enable If the exception is enabled, a Floating-point Round exception is taken if the result of a Floating-point instruction does not result in overflow or underflow, and the result is inexact (FG | FX = 1), or if the result of a Floating-point instruction does result in overflow (FOVF = 1) but Floating-point Overflow exceptions are disabled (FOVFE = 0), or if the result of a Floating-point instruction results in underflow (FUNF = 1 or FUNFH = 1) but Floating-point Underflow exceptions are disabled (FUNFE = 0), and no Floating-point Data exception occurs. 0 Exception disabled 1 Exception enabled
26 (58)	FINVE	Embedded Floating-point Invalid Operation / Input Error Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FINV bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
27 (59)	FDBZE	Embedded Floating-point Divide by Zero Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FDBZ bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
28 (60)	FUNFE	Embedded Floating-point Underflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FUNF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
29 (61)	FOVFE	Embedded Floating-point Overflow Exception Enable If the exception is enabled, a Floating-point Data exception is taken if the FOVF bit is set by a floating-point instruction. 0 Exception disabled 1 Exception enabled
30:31	FRMC	Embedded Floating-point Rounding Mode Control

Table 63-7. EFPU Status and Control Register field descriptions

Bits	Name	Description
(62:63)		00 Round to Nearest 01 Round toward Zero 10 Round toward +Infinity 11 Round toward -Infinity

63.7 Cache

This section describes the cache registers, cache control instructions, and various cache operations.

63.7.1 Cache overview

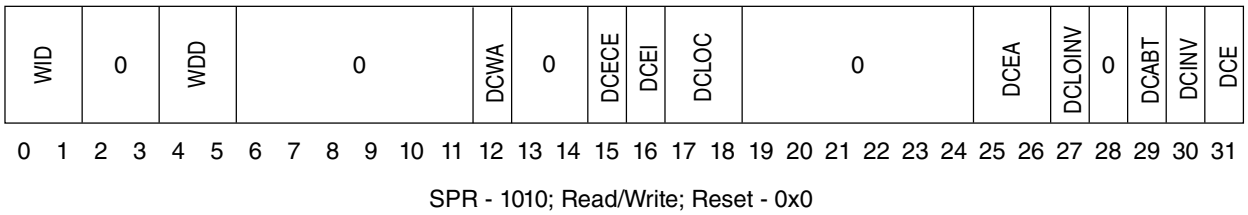
The e200z4204n3 core supports an 8 KB 2-way set-associative instruction cache and a 4 KB 2-way set-associative data cache with a 32-byte line size. The caches improve system performance by providing low-latency data to the e200z4204n3 instruction and data pipelines, which decouples processor performance from system memory performance.

The e200z4204n3 core also contains an 8-entry store buffer to decouple store completion to the processor from store completion on the data interface to further improve system performance.

Instruction and data addresses from the processor are used to index the cache arrays. If the access address matches a valid cache tag entry, the access hits in the cache.

63.7.2 L1 Cache Control and Status Register 0 (L1CSR0)

The L1 Cache Control and Status Register 0 (L1CSR0) is a 32-bit register used for general control of the data cache and for disabling ways in both caches. The L1CSR0 register is accessed using a **mfspr** or **mtspr** instruction. The L1CSR0 register is shown in the following figure.



SPR - 1010; Read/Write; Reset - 0x0

Figure 63-8. L1 Cache Control and Status Register 0 (L1CSR0)

The L1CSR0 fields are described in the following table.

Table 63-8. L1CSR0 field descriptions

Bits	Name	Description
0:1	WID	Way Instruction Disable Bit 0 corresponds to way 0. Bit 1 corresponds to way 1. The WID bits may be used for locking ways of the instruction cache, and also affect the replacement policy of the instruction cache. 0 The corresponding way in the instruction cache is available for replacement by instruction miss line fills. 1 The corresponding way instruction cache is not available for replacement by instruction miss line fills.
2:3	—	Reserved[1]
4:5	WDD	Way Data Disable Bit 4 corresponds to way 0. Bit 5 corresponds to way 1. The WDD bits may be used for locking ways of the data cache, and also affect the replacement policy of the data cache. 0 The corresponding way in the data cache is available for replacement by data miss line fills. 1 The corresponding way in the data cache is not available for replacement by data miss line fills.
6:11	—	Reserved[2]
12	DCWA	Data Cache Write Allocation Policy This bit also controls merging of store data into the linefill buffer while a cache linefill is in progress. Store data will not be merged when write allocation is disabled. 0 Cache line allocation on a cacheable write miss is disabled 1 Cache line allocation on a cacheable write miss is enabled
13:14	—	Reserved[2]
15	DCECE	Data Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	DCEI	Data Cache Error Injection

Table continues on the next page...

Table 63-8. L1CSR0 field descriptions (continued)

Bits	Name	Description
		DCEI will cause injection of errors regardless of the setting of DCECE, although reporting of errors will be masked while DCECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected on each write into the cache data array by inverting the two uppermost parity check bits (p_dchk[0:1]). This includes writes due to store hits as well as writes due to cache line refills.
17:18	DCLOC	Data Cache Lockout Control **Note:** For the **dcbi** and **dcbf** instructions, and for specialized load/store instructions, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled. **Note:** When operating in MC mode (DCEA = 00), detected errors on cache-inhibited accesses will not cause a data cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved[2]
25:26	DCEA	Data Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	DCLOINV	Data Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing DCINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation

Table continues on the next page...

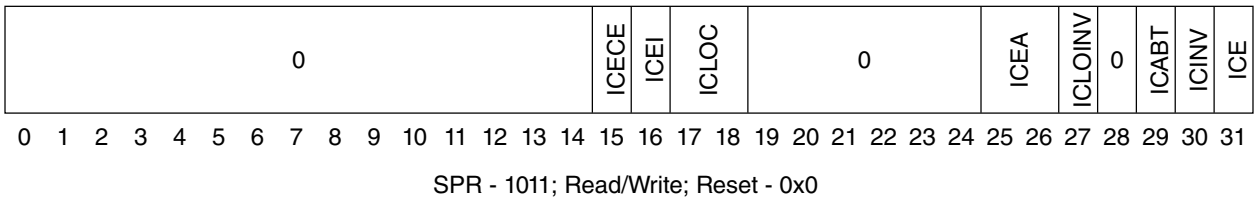
Table 63-8. L1CSR0 field descriptions (continued)

Bits	Name	Description
		operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. When DCINV = 0: Reserved, do not set to 1 When DCINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved[2]
29	DCABT	Data Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.
30	DCINV	Data Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the data cache enable (DCE) value. 0 No cache invalidate 1 Cache invalidation operation
31	DCE	Data Cache Enable When disabled, cache lookups are not performed for normal load or store accesses.Other L1CSR0 cache control operations are still available. Also, operation of the store buffer is not affected by DCE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

63.7.3 L1 Cache Control and Status Register 1 (L1CSR1)

The L1 Cache Control and Status Register 1 (L1CSR1) is a 32-bit register used for general control of the instruction cache. The L1CSR1 register is accessed using a **mfspr** or **mtspr** instruction. The L1CSR1 register is shown in the following figure.



SPR - 1011; Read/Write; Reset - 0x0

Figure 63-9. L1 Cache Control and Status Register 1 (L1CSR1)

The L1CSR1 fields are described in the following table.

Table 63-9. L1CSR1 field descriptions

Bits	Name	Description
0:14	—	Reserved[1]
15	ICECE	Instruction Cache Error Checking Enable 0 Error Checking is disabled 1 Error Checking is enabled
16	ICEI	Instruction Cache Error Injection Enable ICEI will cause injection of errors regardless of the setting of ICECE, although reporting of errors will be masked when ICECE = 0. 0 Cache Error Injection is disabled 1 A double-bit error will be injected into each doubleword written into the cache by inverting the two uppermost parity check bits.
17:18	ICLOC	Instruction Cache Lockout Control **Note:** For the **icbi** instruction, no lockout operation is performed, regardless of the occurrence of EDC/ECC error conditions, and errors are handled in the same manner as when lockout is disabled. See ECC/EDC Error Handling for Cache Control Operations and Instructions. **Note:** When operating in MC mode (ICEA = 00), detected errors on cache-inhibited accesses will not cause an instruction cache line to be locked out, instead the cache line contents are ignored. 00 Cache line lockout is disabled (and array LO indicators are ignored). 01 Cache line lockout is enabled. Cache lines with any tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. 10 Cache line lockout is enabled. Cache lines with any tag or data errors will have the corresponding lockout indicators set. A Machine check will still be generated for an error if the operation would have normally generated one. 11 Cache line lockout is enabled. Cache lines with uncorrectable tag errors or any data errors will have the corresponding lockout indicators set. A Machine check will not be generated for the error if the operation would have normally generated one unless a locked line is locked out. If a correctable tag error occurs, and is re-detected on recycling the access after a correction cycle for the tag is performed, the line is locked out regardless of detecting a single-bit or multi-bit error.
19:24	—	Reserved[1]

Table continues on the next page...

Table 63-9. L1CSR1 field descriptions (continued)

Bits	Name	Description
25:26	ICEA	Instruction Cache Error Action 00 Error Detection causes Machine Check exception. 01 Error Detection causes Correction/Auto-invalidation. No machine check is generated for most cases. Correction is performed for single-bit tag errors, and lines with multi-bit tag errors are invalidated. Correction is performed for single or multi-bit data errors on cache hits by reloading of the line. 10 Reserved 11 Reserved
27	ICLOINV	Instruction Cache Lockout Indicator Invalidate When written to a 1 in conjunction with writing ICINV to a 1, a cache lockout indicator invalidation operation is initiated by hardware, and the LO bits are cleared, removing all line lockout status. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache lockout bit invalidation operations require approximately 66 cycles to complete. Invalidation occurs regardless of the instruction cache enable (ICE) value. When ICINV = 0: Reserved, do not set to 1 When ICINV = 1: 0 No cache lockout bit invalidate 1 Cache lockout indicator invalidation operation
28	—	Reserved[1]
29	ICABT	Instruction Cache Operation Aborted Indicates a Cache Invalidate operation was aborted prior to completion. This bit is set by hardware on an aborted condition, and will remain set until cleared by software writing 0 to this bit location.
30	ICINV	Instruction Cache Invalidate When written to a 1, a cache invalidation operation is initiated by hardware. Once complete, this bit is reset to 0. Writing a 1 while an invalidation operation is in progress will result in an undefined operation. Writing a 0 to this bit while an invalidation operation is in progress will be ignored. Cache invalidation operations require approximately 130 cycles to complete. Invalidation occurs regardless of the enable (ICE) value. 0 No cache invalidate 1 Cache invalidation operation
31	ICE	Instruction Cache Enable When disabled, cache lookups are not performed for instruction accesses. Other L1CSR1 cache control operations are still available and are not affected by ICE. 0 Cache is disabled 1 Cache is enabled

1. These bits are not implemented and should be written with zero for future compatibility.

63.7.4 L1 Cache Configuration Register 0 (L1CFG0)

The L1 Cache Configuration Register 0 (L1CFG0) is a 32-bit read-only register. L1CFG0 provides information about the configuration of the e200z4204n3 L1 data cache design. The contents of the L1CFG0 register can be read using a **mfspr** instruction. The L1CFG0 register is shown in the following figure.

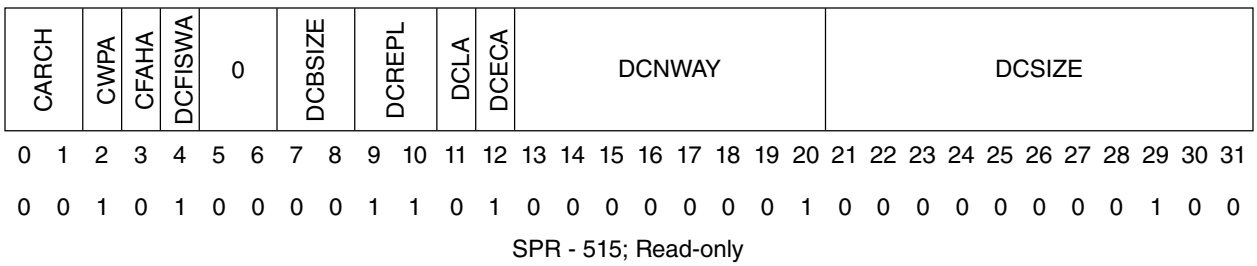


Figure 63-10. L1 Cache Configuration Register 0 (L1CFG0)

The L1CFG0 bits are described in the following table.

Table 63-10. L1CFG0 field descriptions

Bits	Name	Description
0:1	CARCH	Cache Architecture 00 The cache architecture is Harvard
2	CWPA	Cache Way Partitioning Available 1 The caches support partitioning of way availability for I/D accesses
3	DCFAHA	Data Cache Flush All by Hardware Available 0 The data cache does not support Flush All in Hardware
4	DCFISWA	Data Cache Flush/Invalidate by Set and Way Available 1 The data cache supports invalidation by Set and Way via L1FINV0
5:6	—	Reserved - read as zeros
7:8	DCBSIZE	Data Cache Block Size 00 The data cache implements a block size of 32 bytes
9:10	DCREPL	Data Cache Replacement Policy 11 The data cache implements a FIFO replacement policy
11	DCLA	Data Cache Locking APU Available 0 The data cache does not implement the line locking APU
12	DCECA	Data Cache Error Checking Available 1 The data cache implements error checking
13:20	DCNWAY	Data Cache Number of Ways 0x01 The data cache is 2-way set-associative
21:31	DCSIZE	Data Cache Size 0x004 The size of the data cache is 4 KB

63.7.5 L1 Cache Configuration Register 1 (L1CFG1)

The L1 Cache Configuration Register 1 (L1CFG1) is a 32-bit read-only register. L1CFG1 provides information about the configuration of the e200z4204n3 L1 instruction cache design. The contents of the L1CFG1 register can be read using a **mfspr** instruction. The L1CFG1 register is shown in the following figure.

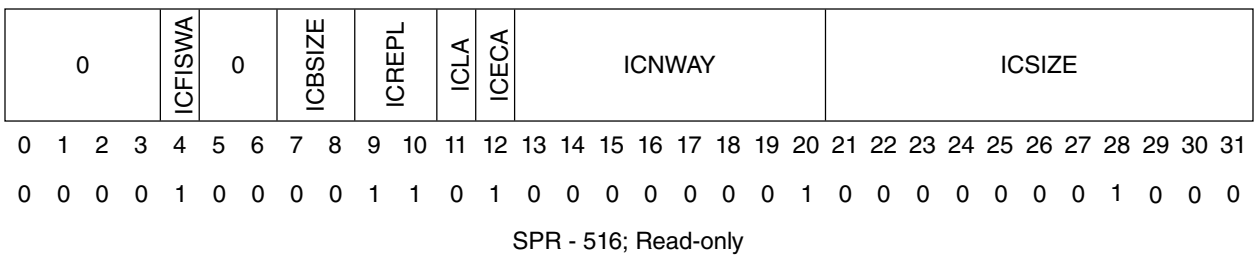


Figure 63-11. L1 Cache Configuration Register 1 (L1CFG1)

The L1CFG1 fields are described in the following table.

Table 63-11. L1CFG1 field descriptions

Bits	Name	Description
0:3	—	Reserved - read as zeros
4	ICFISWA	Instruction Cache Flush/Invalidate by Set and Way Available 1 The instruction cache supports invalidation by Set and Way via L1FINV1
5:6	—	Reserved - read as zeros
7:8	ICBSIZE	Instruction Cache Block Size 00 The instruction cache implements a block size of 32 bytes
9:10	ICREPL	Instruction Cache Replacement Policy 11 The instruction cache implements a FIFO replacement policy
11	ICLA	Instruction Cache Locking APU Available 0 The instruction cache does not implement the line locking APU
12	ICECA	Instruction Cache Error Checking Available 1 The instruction cache implements error checking
13:20	ICNWAY	Instruction Cache Number of Ways 0x01 The instruction cache is 2-way set-associative
21:31	ICSIZE	Instruction Cache Size 0x008 The size of the instruction cache is 8 KB

63.7.6 Data cache software coherency

Data cache coherency is supported through software operations to invalidate lines using either a **dcbi** or **dcbf** instruction, or by using the L1FINV0 control register.

Data cache misses will force the store buffer to empty prior to performing the access.

63.7.7 Data cache hardware coherency

Data cache coherency is not supported in hardware. Software operations must generally be used to maintain coherency. Debug support is provided, however, for a D-Cache line invalidation operation requested by an external hardware debugger. When requested by an external hardware debugger tool operating through the OnCE port, an individual cache line invalidation operation will be scheduled to occur at the next available D-Cache access cycle.

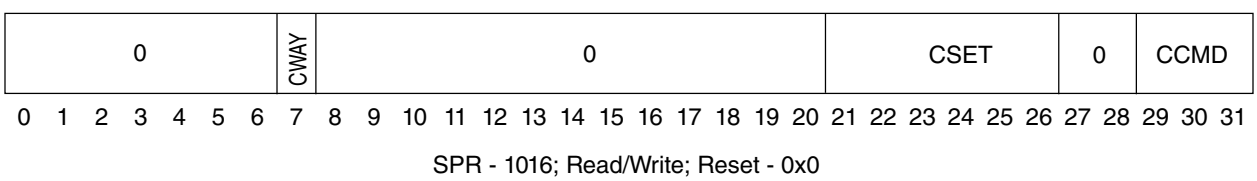
63.7.8 Cache invalidate by set and way

The e200z4204n3 supports cache set/way invalidation under software control. The caches may be invalidated by index and way through a **mtspr l1finv{0,1}** instruction.

The L1 Flush and Invalidate Control Registers (L1FINV{0.1}) are 32-bit SPRs used to select a cache set and way to be invalidated. This function is available even when a cache is disabled. L1FINV0 is used for data cache operations, while L1FINV1 is used for instruction cache operations.

63.7.8.1 L1 Flush and Invalidate Control Register 0 (L1FINV0)

The SPR number for L1FINV0 is 1016 in decimal. The L1FINV0 register is shown in the following figure.



SPR - 1016; Read/Write; Reset - 0x0

Figure 63-12. L1 Flush/Invalidate Register 0 (L1FINV0)

The L1FINV0 bits are described in the following table.

Table 63-12. L1FINV0 field descriptions

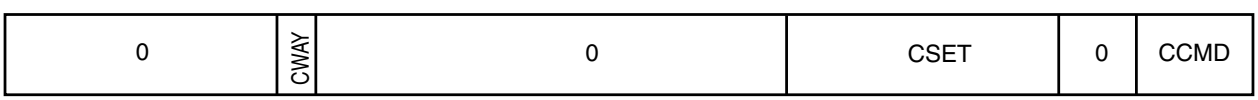
Bits	Name	Description
0:6	—	Reserved[1] for way extension
7	CWAY	Cache Way Specifies the data cache way to be selected
8:20	—	Reserved[1] for set extension
21:26	CSET	Cache Set Specifies the cache set to be selected
27:28	—	Reserved[1] for set/command extension
29:31	CCMD	Cache Command 000 = The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV0, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

63.7.8.2 L1 Flush and Invalidate Control Register 1 (L1FINV1)

The L1FINV1 register is shown in following figure.



SPR - 959; Read/Write; Reset - 0x0

Figure 63-13. L1 Flush/Invalidate Register 1 (L1FINV1)

The L1FINV1 bits are described in the following table.

Table 63-13. L1FINV1 field descriptions

Bits	Name	Description
0:6	—	Reserved for way extension
7	CWAY	Cache Way

Table continues on the next page...

Table 63-13. L1FINV1 field descriptions (continued)

Bits	Name	Description
		Specifies the instruction cache way to be selected
8:19	—	Reserved[1] for set extension
20:26	CSET	Cache Set Specifies the instruction cache set to be selected
27:28	—	Reserved[1] for set/command extension
29:31	CCMD	Cache Command 000 The data contained in this entry is invalidated. LO bits are unaffected 001 Reserved 01x Reserved 100 Reserved 101 The data contained in this entry is invalidated. LO bits are cleared 11x Reserved

1. These bits are not implemented and should be written with zero for future compatibility.

For invalidation operations via L1FINV1, tag ECC errors and data EDC errors are ignored, and the selected line will be invalidated regardless of any error. Invalidations conditionally affect the state of the lockout bits (LO).

63.7.9 Cache EDC/ECC parity protection

Cache parity protection is supported for both the tag and data arrays of each cache. Seven parity check bits are provided for each tag entry for the tag arrays of both caches to support error detection and correction (ECC - SECDED). Eight parity check bits are provided for each doubleword in the data arrays of the I-Cache, and each word in the data arrays of the D-Cache, which are used for single- and double-bit error detection (EDC - DED, double error detection). Utilizing ECC/EDC protection, many multi-bit errors are also detected.

The error detection codes also cover the cache index address of the cache line, providing additional protection against addressing errors. If any type of error (including a single-bit error in one of the index bits) is encountered in one of the index address bits, it is treated as an uncorrectable tag ECC error to protect against internal addressing failures. No attempt will be made to correct single-bit errors where the syndrome indicates an index address bit.

Tag ECC and data EDC checking is controlled by the L1CSR0[DCECE] and L1CSR1[ICECE] control fields. When error checking is enabled, checking is performed on each cache access. Errors are not signaled by the respective cache when cache error checking is disabled for that cache (L1CSR $\{0, 1\}_{[D,I]CECE} = 0$).

If an uncorrectable tag ECC error is detected on any portion of a tag accessed during cache lookups performed because of instruction fetching, normal loads, or normal stores, a tag ECC error is signaled, regardless of whether a cache hit or miss occurs. Otherwise, if a cache hit for an instruction fetch or a normal load occurs and a data EDC error is detected on any portion of the accessed data, a parity error is also signaled.

Store hits to the D-Cache will be placed into one of the RMW buffers to support EDC checkbit operation. If the store is a partial-width store, the cache data word corresponding to the address of the store is read into the next available buffer, the store data is merged, and the new EDC checkbits for the word are calculated. If the store is a full-width store, no read of the cache data is performed, since the store will overwrite the full EDC data granularity, and the EDC checkbits can be computed directly. Buffers are managed on a FIFO basis. If no buffer is available to hold the store data, a stall is incurred while a buffer (or two if possible) is written back to the cache to be freed for holding the new store. Buffers are written back to the D-Cache during otherwise idle cycles when two buffers are occupied, providing a simple form of store gathering.

Signaling of an ECC error or EDC error may cause a Machine Check exception to occur. One or more syndrome bits may be set in the Machine Check Syndrome register, or may instead result in a correction/auto-invalidation operation and not result in an exception being signaled. Both may occur, depending on the error action control setting in the appropriate cache control register.

63.7.9.1 Cache line lockout

In addition to the ECC/EDC protection employed for the caches, each cache line in the I-Cache and D-Cache has a lockout indicator composed of a redundant set of lockout bits. These lockout bits can selectively be set when certain errors occur in either the tag or the data portion of the cache line. Use of the lockout function and control over error conditions that cause a lockout to occur are controlled by $L1CSR\{0,1\}_{[D,I]CLOC}$. When the lockout function is enabled, lines that encounter selected tag ECC or data EDC errors on normal instruction fetch. load. or store accesses will have their lockout bits set. When the lockout indicator is set, the line will not be replaced and will remain in an invalid state, effectively disabling it. Also, future tag ECC or data EDC errors on the line are ignored. Cache-inhibited accesses will only set lockout indicators on lines not in a locked way. In addition, no lockout indicators are set by cache-inhibited accesses when operating in machine check mode.

When operating in machine check mode, if lockout controls are enabled via $L1CSR\{0,1\}_{[D.I]CLOC]}$, lockout bit parity errors on any line detected on a cacheable cache lookup will generate a machine check exception.

If correction/auto-invalidation is instead enabled, on each cache lookup operation for an instruction fetch or normal load or store access, if a single- or double-bit lockout bit parity error is detected in one or more ways and lockout controls are enabled, the lockout error(s) will be corrected by rewriting all lockout bits to the asserted state, and no machine check is generated, unless the line is in a locked way. If a line in a locked way incurs a LO bit parity error, a machine check will be generated to ensure that any possible new lockout of the line is reported. For non-cacheable accesses, lockout bit parity errors will only be corrected for lines not in a locked way. Lockout bit parity errors do not generate a machine check for non-cacheable accesses in either error action mode, thus lockout bit correction is not performed for lockout bit parity errors detected on a line in a locked way.

In addition, to avoid certain exception conditions and for consistency with error reporting, specialized load/store accesses do not set LO bits; instead, cache contents are ignored for lines with those errors. Cache flush and invalidate instructions do not report or correct lockout bit parity errors. For both of these cases, a future cacheable normal access will perform the lockout function if required.

63.7.10 Cache error injection

Cache error injection provides a way to test error recovery by intentionally injecting parity errors into the instruction and/or data cache.

Error injection into the instruction cache operates as follows:

- If $L1CSR1_{ICEI}$ is set, any instruction cache line fill to the instruction cache data has the associated two most significant parity check bits inverted in the instruction cache data array for each doubleword loaded.

Error injection for the data cache operates as follows:

- If $L1CSR0_{DCEI}$ is set, any cache line fill to the data cache data array has the associated two most significant parity check bits inverted in the data array for each word loaded. Additionally, inverted parity bits are generated for any data stored into the data cache data array on a store hit.

Cache parity error injection is not performed for cache debug write accesses, since parity bit values written can be directly controlled.

In order to clear the parity errors, a cache invalidation or an invalidation of the lines that could have had an injected parity error may be performed. Line invalidation may be performed by an **icbi/dcbi** instruction or an L1FINV{0.1} invalidation operation.

63.8 Exceptions

Interrupts implemented in the e200z4204n3 core and the exception conditions that cause them are listed in the following table.

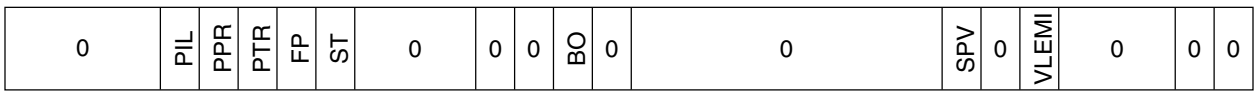
Table 63-14. Exceptions and conditions

Interrupt type	Interrupt vector offset value	Causing conditions
System reset	none, vector to [p_rstbase[0:29]] || 2'b00	Reset
Critical Input	0x00[1]	**p_critint_b** is asserted and MSR_{CE} = 1.
Machine check	0x10	1. **p_mcp_b** transitions from negated to asserted 2. ISI or Bus Error on first instruction fetch for an exception handler 3. Parity Error signaled on Cache access 4. External bus error 5. Stack limit check failure
Machine check (NMI)	0x10	Non-Maskable Interrupt
Data Storage	0x20	Access control
Instruction Storage	0x30	Access control
External Input	0x40[2]	Interrupt Controller interrupt and MSR_{EE} = 1
Alignment	0x50	1. **lmw**, **stmw** not word aligned 2. **lwarx** or **stwcx.** not word aligned, **lharx** or **sthcx.** not halfword aligned 3. **dcbz**
Program	0x60	Illegal, Privileged, Trap
Performance Monitor	0x70	Performance Monitor Enabled Condition or Event w/PMGC0[UDI] = 0
System call	0x80	Execution of the System Call (**se_sc**) instruction
Debug	0x90	Trap, Instruction Address Compare, Data Address Compare, Instruction Complete, Branch Taken, Return from Interrupt, Interrupt Taken, Debug Counter, External Debug Event, Unconditional Debug Event, Performance Monitor Enabled Condition or Event w/ $PMGC0_{UDI}$ = 1
EFPU Data Exception	0xA0	Embedded Floating-point Data Exception
EFPU Round Exception	0xB0	Embedded Floating-point Round Exception
TBD	0xC0–0xF0	Reserved for future processor use

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

63.8.1 Exception Syndrome Register (ESR)

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



SPR - 62; Read/Write; Reset - 0x0

Figure 63-14. Exception Syndrome Register (ESR)

The ESR bits are defined in the following table.

Table 63-15. ESR field descriptions

Bit(s)	Name	Description	Associated interrupt type
0:3	—	Reserved	—
4	PIL	Illegal Instruction exception For e200z4204n3, PIL is used for both illegal and unimplemented instructions.	Program
5	PPR	Privileged Instruction exception	Program
6	PTR	Trap exception	Program
7	FP	Floating-point operation	Program
8	ST	Store operation	Alignment Data Storage
9:11	—	Reserved	—
12	—	Reserved	—
13	—	Reserved	—
14	BO	Byte Ordering exception Mismatched Instruction Storage exception	Instruction Storage
15	—	Reserved	—
16:23	—	Reserved	—
24	SPV	EFPU APU Operation	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Alignment Data Storage
25	—	Reserved	—
26	VLEMI	VLE Mode Instruction	EFPU Floating-point Data Exception EFPU Floating-point Round Exception Data Storage Instruction Storage

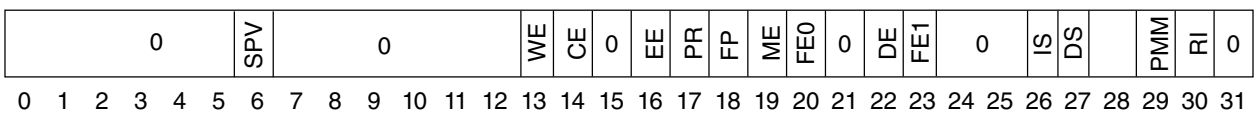
Table continues on the next page...

Table 63-15. ESR field descriptions (continued)

Bit(s)	Name	Description	Associated interrupt type
			Alignment Program System Call
27:29	—	Reserved	—
30	—	Reserved	—
31	—	Reserved	—

63.8.2 Machine State Register (MSR)

The Machine State Register defines the state of the processor. The e200z4204n3 MSR is shown in the following figure.



Read/Write; Reset - 0x0

Figure 63-15. Machine State Register (MSR)

The MSR bits are defined in the following table.

Table 63-16. MSR field descriptions

Bit(s)	Name	Description
0:5	—	Reserved
6	SPV	SP/Embedded FP/Vector available 0 SP/Embedded FP Double/Vector unit is unavailable. The processor cannot execute SP/ Embedded FP Double or Vector Unit instructions. 1 SP/Embedded FP Double/Vector unit is available. The processor can execute SP/ Embedded FP Double or Vector Unit instructions. **NOTE:** For e200z4204n3, these units are not supported in hardware, and an Illegal Instruction exception will be generated for attempted execution of these instructions regardless of the setting of SPV. SPV is ignored, but cleared on exceptions.
7:12	—	Reserved
13	WE	Wait State (Power management) enable. **NOTE:** This bit is implementation dependent and is being phased out. It will be removed in a future release. Currently it is implemented as a writeable bit, and is ignored, but cleared on exceptions.
14	CE	Critical Interrupt Enable 0 Critical Input interrupts are disabled. 1 Critical Input interrupts are enabled.

Table continues on the next page...

Table 63-16. MSR field descriptions (continued)

Bit(s)	Name	Description
15	—	Reserved
16	EE	External Interrupt Enable 0 External Input interrupts are disabled. 1 External Input interrupts are enabled.
17	PR	Problem State 0 The processor is in supervisor mode, can execute any instruction, and can access any resource (e.g. GPRs, SPRs, MSR, etc.). 1 The processor is in user mode, cannot execute any privileged instruction, and cannot access any privileged resource.
18	FP	0 Floating point unit is unavailable. The processor cannot execute floating-point instructions, including floating-point loads, stores, and moves. 1 Floating Point unit is available. The processor can execute floating-point instructions. **NOTE:** For e200z4204n3, the PowerISA 2.06 floating point unit is not supported in hardware, and an Illegal Instruction exception will be generated for attempted execution of PowerISA 2.06 floating point instructions regardless of the setting of FP. FP is ignored, but cleared on exceptions.
19	ME	Machine Check Enable 0 Asynchronous Machine Check interrupts are disabled. 1 Asynchronous Machine Check interrupts are enabled.
20	FE0	Floating-point exception mode 0 (not used by Zen) **NOTE:** for e200z4204n3, the PowerISA 2.06 floating point unit is not supported in hardware, thus FE0 is ignored, but cleared on exceptions.
21	—	Reserved
22	DE	Debug Interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.
23	FE1	Floating-point exception mode 1 (not used by Zen) **NOTE:** For e200z4204n3, the PowerISA 2.06 floating point unit is not supported in hardware, thus FE1 is ignored, but cleared on exceptions.
24:25	—	Reserved
26	IS	Instruction Address Space 0 The processor directs all instruction fetches to address space 0. 1 The processor directs all instruction fetches to address space 1. **NOTE:** For e200z4204n3, Address Spaces are not supported, thus the IS bit is ignored, but cleared on exceptions.
27	DS	Data Address Space 0 The processor directs all data storage accesses to address space 0. 1 The processor directs all data storage accesses to address space 1. **NOTE:** For e200z4204n3, Address Spaces are not supported, thus the DS bit is ignored, but cleared on exceptions.
28	—	Reserved

Table continues on the next page...

Table 63-16. MSR field descriptions (continued)

Bit(s)	Name	Description
29	PMM	Performance monitor mark bit System software can set PMM when a marked process is running to enable statistics to be gathered only during the execution of the marked process. MSR_{PR} and MSR_{PMM} together define a state that the processor (supervisor or user) and the process (marked or unmarked) may be in at any time. If this state matches an individual state specified in the PMLCa*n* Performance Monitor registers, the state for which monitoring is enabled, counting is enabled.
30	RI	Recoverable Interrupt This bit is provided for software use to detect nested machine check exception conditions. This bit is cleared by hardware when a Machine Check interrupt is taken.
31	—	Reserved

63.8.3 Machine Check Syndrome Register (MCSR)

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

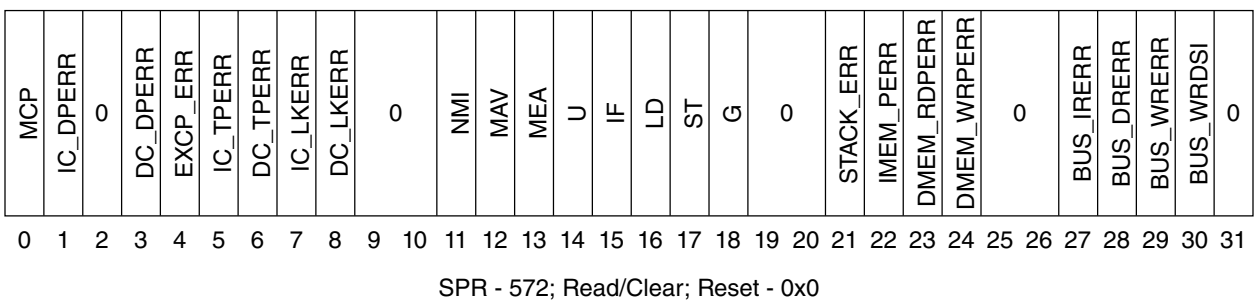


Figure 63-16. Machine Check Syndrome Register (MCSR)

The following table describes MCSR fields. The MCSR indicates the source of a machine check condition.

All bits in the MCSR are implemented as "write 1 to clear." Software in the machine check handler is expected to clear the MCSR bits it has sampled prior to re-enabling MSR[ME] to avoid a redundant machine check exception and to prepare for updated status bit information on the next machine check interrupt.

Note that any set bit in the MCSR other than status-type bits will cause a subsequent machine check interrupt once MSR[ME] = 1.

Table 63-17. Machine Check Syndrome Register (MCSR) field descriptions

Bit	Name	Description	Exception type[1]	Recoverable
0	MCP	Machine check input pin	Async Mchk	Maybe
1	IC_DPERR	Instruction Cache data array parity error	Async Mchk	Precise
2	—	Reserved	—	—
3	DC_DPERR	Data Cache data array parity error	Async Mchk	Maybe
4	EXCP_ERR	ISI or Bus Error on first instruction fetch for an exception handler	Async Mchk	Precise
5	IC_TPERR	Instruction Cache Tag parity error	Async Mchk	Precise
6	DC_TPERR	Data Cache Tag parity error	Async Mchk	Maybe
7	IC_LKERR	Instruction Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the I-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
8	DC_LKERR	Data Cache Lock error Indicates a cache control operation or invalidation operation invalidated one or more lines in a locked way of the D-Cache for certain situations. May also be set on locked line refill error.	Async Mchk	—
9:10	—	Reserved	—	—
11	NMI	NMI input pin	NMI	—
12	MAV	MCAR Address Valid Indicates that the address contained in the MCAR was updated by hardware to correspond to the first detected Async Mchk error condition	Status	—
13	MEA	MCAR holds Effective Address If MAV = 1, MEA = 1 indicates that the MCAR contains an effective address and MEA = 0 indicates that the MCAR contains a physical address[2]	Status	—
14	U	User Indicates the value captured in MCAR was generated in user mode.	Status	—
15	IF	Instruction Fetch Error Report An error occurred during the fetch of an instruction and the instruction attempted to execute. This could be due to an internal parity error, or an external bus error. MCSRR0 contains the instruction address.	Error Report	Precise
16	LD	Load type instruction Error Report An error occurred during the attempt to execute the load type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or an external bus error.	Error Report	Precise
17	ST	Store type instruction Error Report	Error Report	Precise

Table continues on the next page...

Table 63-17. Machine Check Syndrome Register (MCSR) field descriptions (continued)

Bit	Name	Description	Exception type[1]	Recoverable
		An error occurred during the attempt to execute the store type instruction located at the address stored in MCSRR0. This could be due to an internal parity error, stack limit check error, or on certain external bus errors.		
18	G	Guarded instruction Error Report An error occurred during the attempt to execute the load or store type instruction located at the address stored in MCSRR0 and the access was guarded and encountered an error on the external bus.	Error Report	Precise
19:20	—	Reserved	—	—
21	STACK_ERR	Stack Access Limit Check Error Indicates a limit check failure on a CPU data access using R1 in the <EA> calculation.	Async Mchk	Precise
22	IMEM_PERR[3]	Instruction Mem (IMEM) Parity Error Indicates an uncorrectable error in the IMEM on a CPU port access.	Async Mchk	Precise
23	DMEM_RDPERR[4]	Data Mem (DMEM) Parity Error Indicates an uncorrectable error in the DMEM on a CPU port read access.	Async Mchk	Precise
24	DMEM_WRPERR	Data Mem (DMEM) Write Parity Error Indicates an uncorrectable error in the DMEM on a CPU port write access.	Async Mchk	Maybe
25:26	—	Reserved	—	—
27	BUS_IRERR	Read bus error on Instruction recovery linefill	Async Mchk	Precise if data used
28	BUS_DRERR	Read bus error on data load or linefill	Async Mchk	Precise if data used
29	BUS_WRERR	Write bus error on store to bus	Async Mchk	Unlikely
30	BUS_WRDSI	Write bus error on buffered store to bus with DSI signaled. Set concurrently with BUS_WRERR for this case	Async Mchk	Unlikely
31	—	Reserved	—	—

1. Indicates the exception type associated with a given syndrome bit.
   - "Error Report" indicates that this bit is only set for error report exceptions which cause machine check interrupts. These bits are only updated when the machine check interrupt is actually taken. Error report exceptions are not gated by MSR[ME]. These are synchronous exceptions. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.
   - "Status" indicates that this bit is provides additional status information regarding the logging of a machine check exception. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.

- "NMI" indicates that this bit is only set for the non-maskable interrupt type exception which causes a machine check interrupt. This bit is only updated when the machine check interrupt is actually taken. NMI exceptions are not gated by MSR[ME]. This is an asynchronous exception. This bit will remain set until cleared by software writing a '1' to the bit position.
- "Async Mchk" indicates that this bit is set for an asynchronous machine check exception. These bits are set immediately upon detection of the error. Once any "Async Mchk" bit is set in the MCSR, a machine check interrupt will occur if MSR[ME]= 1. If MSR[ME] = 0, the machine check exception will remain pending. These bits will remain set until cleared by software writing a '1' to the bit position(s) to be cleared.

2. Note that since no address translation mechanism is present on e200z4204n3, the MEA value is always set to 0.
3. Will not be set by e200z4204n3 since no local instruction memory is present.
4. Note that since no local data memory is present on e200z4204n3.

63.8.4 Interrupt Vector Prefix Registers (IVPR)

The Interrupt Vector Prefix Register is used during interrupt processing for determining the starting address of a software handler used to handle an interrupt. The value of the Vector Offset selected for a particular interrupt type is concatenated with the Vector Base value held in the IVPR to form an instruction address from which execution is to begin. The format of IVPR is shown in the following figure.

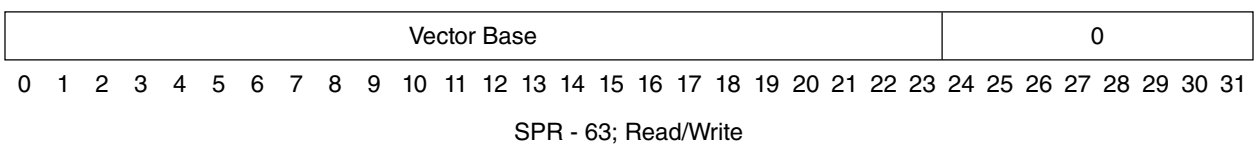


Figure 63-17. Interrupt Vector Prefix Register (IVPR)

The IVPR fields are defined in the following table.

Table 63-18. IVPR field descriptions

Bit(s)	Name	Description
0:23 (32:55)	Vector Base	Vector Base This field is used to define the base location of the vector table, aligned to a 256-byte boundary. This field provides the high-order 24 bits of the location of all interrupt handlers (unless hardware vectoring is used). The vector offset value appropriate for the type of exception being processed is concatenated with the IVPR Vector Base to form the address of the handler in memory. For hardware-vectored interrupts, the value of the supplied interrupt vector is logically ORed with the low order bits of the Vec Base Field to determine the interrupt handler address.
24:31 (56:63)	—	Reserved

63.8.5 Interrupt definitions

63.8.5.1 Critical Input Interrupt (Offset 0x00)

A Critical Input exception is signalled to the processor by the assertion of the critical interrupt pin (**p_critint_b**). When e200z4204n3 detects the exception, if the exception is enabled by MSRCE, the core takes the Critical Input interrupt. The **p_critint_b** input is a level-sensitive signal expected to remain asserted until the core acknowledges the interrupt. If **p_critint_b** is negated early, recognition of the interrupt request is not guaranteed. After the core begins execution of the critical interrupt handler, the system can safely negate **p_critint_b**.

A Critical Input interrupt may be delayed by other higher priority exceptions or if MSR_{CE} is cleared when the exception occurs.

The following table lists register settings when a Critical Input interrupt is taken.

Table 63-19. Critical Input Interrupt - Register Settings

Register	Setting Description		
CSRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.		
CSRR1	Set to the contents of the MSR at the time of the interrupt		
MSR	SPV 0 WE 0 CE 0 EE 0 PR 0	FP 0 ME — FE0 0 DE —/0[1]	FE1 0 IS 0 DS 0 PMM 0 RI —
ESR	Unchanged		
MCSR	Unchanged		
DEAR	Unchanged		
Vector	$IVPR_{0:23}$ || 0x00 (autovectored) $IVPR_{0:15}$ || ($IVPR_{16:29}$ | **p_voffset[0:13])** || 2‘b00 (non-autovectored)		

1. Clearing of DE is optionally supported by control in HID0.

The MSR_{DE} bit is not automatically cleared by a Critical Input interrupt, but can be configured to be cleared via the HID0 register ($HID0_{CICLRDE}$).

0x00 is the vector offset used by autovectored Critical Input interrupts to determine the interrupt handler location. e200z4204n3 also provides the capability to directly vector Critical Input interrupts to multiple handlers by allowing a Critical Input interrupt request to be accompanied by a vector offset. The **p_voffset[0:13]** input signals are used to form the interrupt vector when a Critical Input interrupt request is not autovectored (**p_avec_b** negated when **p_critint_b** asserted).

63.8.5.2 Machine Check interrupt (offset 0x10)

The EIS Machine Check APU defines a separate set of save/restore registers (MCSRR0/1), a Machine Check Syndrome Register (MCSR) to record the source(s) of machine checks, and a Machine Check Address Register (MCAR) to hold an address associated with a machine check for certain classes of machine checks. Return from Machine Check instructions (**se_rfmci**) are also provided to support returns using MCSRR0/1.

The MSR[DE] bit is not automatically cleared by a Machine Check exception, but it can be configured to be cleared or left unchanged via the HID0 register (HID0[MCCLRDE]).

When a Machine Check interrupt is taken, registers are updated as shown in the following table.

Table 63-20. Machine Check interrupt—register settings

Register	Setting description					
MCSRR0	On a best-effort basis, set to the address of some instruction that was executing or about to be executing when the machine check condition occurred.					
MCSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0
	CE	0	FE0	0	DS	0
	EE	0	DE	0/—[1]	PMM	0
	PR	0			RI	0
ESR	Unchanged					
MCSR	Updated to reflect the source(s) of a machine check. Hardware only sets appropriate bits, no previously set bits are cleared by hardware.					
Vector	IVPR[0:23] || 0x10					

1. Clearing of DE is optionally supported by control in HID0.

63.8.5.3 Data Storage interrupt (offset 0x20)

A Data Storage interrupt (DSI) may occur if no higher priority exception exists and a Read or Write Access Control exception condition exists.

The following table lists register settings when a DSI is taken.

Table 63-21. Data Storage interrupt—register settings

Register	Setting description
SRR0	Set to the effective address of the excepting load/store instruction.
SRR1	Set to the contents of the MSR at the time of the interrupt

Table continues on the next page...

Table 63-21. Data Storage interrupt—register settings (continued)

Register	Setting description					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Access:		[ST], [SPV], VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	For Access Control exceptions, set to the effective address of the access that caused the violation.					
Vector	IVPR[0:23] || 0x20					

63.8.5.4 Instruction Storage interrupt (offset 0x30)

An Instruction Storage interrupt (ISI) occurs when no higher priority exception exists and an Execute Access Control exception occurs.

The following table lists register settings when an ISI is taken.

Table 63-22. Instruction Storage interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	IVPR[0:23] || 0x30					

63.8.5.5 External Input interrupt (offset 0x40)

An External Input exception is signaled to the processor by the assertion of an interrupt from the interrupt controller. The input is a level-sensitive signal expected to remain asserted until the core acknowledges the external interrupt. If the input is negated early, recognition of the interrupt request is not guaranteed. When the core detects the exception, if the exception is enabled by MSR[EE], it takes the External Input interrupt.

An External Input interrupt may be delayed by other higher priority exceptions or if MSR[EE] is cleared when the exception occurs.

The following table lists register settings when an External Input interrupt is taken.

Table 63-23. External Input interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	IVPR[0:23] || 0x40 (autovectored) IVPR[0:15] || (IVPR[16:29] | **p_voffset[0:13]**) || 2'b00 (non-autovectored)					

63.8.5.6 Alignment interrupt (offset 0x50)

An Alignment exception is generated when any of the following occurs:

- The operand of **lmw** or **stmw** is not word aligned.
- The operand of **lwarx** or **stwcx.** is not word aligned.
- The operand of **lharx** or **sthcx.** is not halfword aligned.
- Execution of a **dcbz** instruction is attempted.

The following table lists register settings when an alignment interrupt is taken.

Table 63-24. Alignment interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting load/store/**dcbz** instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	[ST], VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Set to the effective address of a byte of the load or store access causing the violation.					
Vector	IVPR[0:23] || 0x50					

63.8.5.7 Program interrupt (offset 0x60)

A program interrupt occurs when no higher priority exception exists and one or more of the following exception conditions occur:

- Illegal Instruction exception
- Privileged Instruction exception
- Trap exception

The e200z4204n3 core will invoke an Illegal Instruction program exception on attempted execution of the following instructions:

- Unimplemented instructions
- An instruction from the illegal instruction class
- **mtspr** and **mfspr** instructions with an undefined SPR specified
- **mtdcr** and **mfdcr** instructions with an undefined DCR specified

The e200z4204n3 core will invoke a Privileged Instruction program exception on attempted execution of the following instructions when MSR[PR] = 1 (user mode):

- A privileged instruction
- **mtspr** and **mfspr** instructions that specify a SPRN value with SPRN[5] = 1 (even if the SPR is undefined)

The e200z4204n3 core will invoke an Trap exception on execution of the **tw** instruction if the trap conditions are met and the exception is not also enabled as a Debug interrupt.

The e200z4204n3 core will invoke an Illegal instruction program exception on attempted execution of the instructions **lswi**, **lswx**, **stswi**, **stswx**, **mfapidi**, **mfdcrx**, **mtdcrx**, or on any PowerISA 2.0 floating-point instruction. All other defined or allocated instructions that are not implemented by core will cause an illegal instruction program exception.

The following table lists register settings when a Program interrupt is taken.

Table 63-25. Program interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	Illegal, Unimplemented:		PIL, VLEMI. All other bits cleared.			
	Privileged:		PPR, VLEMI. All other bits cleared.			
	Trap:		PTR, VLEMI. All other bits cleared.			
MCSR	Unchanged					
DEAR	Unchanged					
Vector	IVPR[0:23] || 0x60					

63.8.5.8 Performance Monitor interrupt (offset 0x70)

A performance monitor interrupt can be generated by an enabled condition or event. An enabled condition or event is as follows:

A PMC*x* register overflow condition occurs with the following settings:

- PMLCa*x*[CE] = 11; that is, for the given counter the overflow condition is enabled.
- PMC*x*[OV] = 11; that is, the given counter indicates an overflow.

For a performance monitor interrupt to be signaled on an enabled condition or event, PMGC0[PMIE] must be set.

Although an exception condition may occur with MSR[EE] = 0, the interrupt cannot be taken until MSR[EE] = 1.

The priority of the performance monitor interrupt is below all other asynchronous interrupts.

The following table lists register settings when an performance monitor interrupt is taken.

Table 63-26. Performance Monitor interrupt—register settings

Register	Setting description					
SRR0/ DSRR0[1]	Set to the effective address of the next instruction to be executed.					
SRR1/ DSRR1[1]	Set to the contents of the MSR at the time of the interrupt.					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2]	FE0	0	DS	0
	EE	0/—[3]	DE	—/0[4]	PMM	0
	PR	0			RI	—
ESR	Unchanged					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	IVPR[0:23] || 0x70					

1. DSRR0/1 are used if $PMGC0_{UDI}$ = 1
2. CE is cleared if PMGC0[UDI] = 1 and HID0[DCLRCE] = 1
3. EE is not cleared if PMGC0[UDI] = 1 and HID0[DCLREE] = 0
4. DE is cleared if PMGC0[UDI] = 1

63.8.5.9 System Call interrupt (offset 0x80)

A System Call interrupt occurs when a System Call (**se_sc**) instruction is executed and no higher priority exception exists.

The following table lists register settings when a System Call interrupt is taken.

Table 63-27. System Call interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction *following* the system call instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—

Table continues on the next page...

Table 63-27. System Call interrupt—register settings (continued)

Register	Setting description
ESR	VLEMI. All other bits cleared.
MCSR	Unchanged
DEAR	Unchanged
Vector	IVPR[0:23] || 0x80

63.8.5.10 Debug interrupt (offset 0x90)

Multiple sources can signal a Debug exception. A Debug interrupt occurs when no higher priority exception exists, a Debug exception exists in the Debug Status Register, and Debug interrupts are enabled (both DBCR0[IDM] = 1 (internal debug mode) and MSR[DE] = 1).

The following table lists register settings when a Debug interrupt is taken.

Table 63-28. Debug interrupt—register settings

Register	Setting description					
DSRR0[1]	Set to the effective address of the excepting instruction for IAC, BRT, RET, CRET, and TRAP.Set to the effective address of the next instruction to be executed *following* the excepting instruction for DAC (usually) and ICMP. For UDE, IRPT, CIRPT, DCNT, or DEVT type exceptions, set to the effective address of the instruction that the processor would have attempted to execute next if no exception conditions were present.					
DSRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—/0[2]	FE0	0	DS	0
	EE	—/0[2]	DE	0	PMM	0
	PR	0			RI	—
DBSR[3]	Unconditional Debug Event:		UDE			
	Instr. Complete Debug Event:		ICMP			
	Branch Taken Debug Event:		BRT			
	Interrupt Taken Debug Event:		IRPT			
	Critical Interrupt Taken Debug Event:		CIRPT			
	Trap Instruction Debug Event:		TRAP			
	Instruction Address Compare:		{IAC}			
	Data Address Compare:		{DACR, DACW}			
	Debug Notify Interrupt:		DNI			
	Return Debug Event:		RET			
	Critical Return Debug Event:		CRET			

Table continues on the next page...

Table 63-28. Debug interrupt—register settings (continued)

Register	Setting description	
	External Debug Event: Performance Monitor Debug Event: and optionally, an Imprecise Debug Event flag	{DEVT1, DEVT2} PMI {IDE}
ESR	Unchanged	
MCSR	Unchanged	
DEAR	Unchanged	
Vector	IVPR[0:23] || 0x90	

1. Assumes that the Debug interrupt is precise
2. Conditional based on control bits in HID0
3. Note that multiple DBSR bits may be set

63.8.5.11 Embedded Floating-point Data interrupt (offset 0xA0)

The Embedded Floating-point Data interrupt is taken if no higher priority exception exists and an EFPU Floating-point Data exception is generated. When a Floating-point Data exception occurs, the processor suppresses execution of the instruction causing the exception.

The following table lists register settings when an EFPU Floating-point Data interrupt is taken.

Table 63-29. Embedded Floating-point Data interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	IVPR[0:23] || 0xA0					

63.8.5.12 Embedded Floating-point Round interrupt (offset 0xB0)

The Embedded Floating-point Round interrupt is taken when an EFPU floating-point instruction generates an inexact result and inexact exceptions are enabled.

The following table lists register settings when an EFPU Floating-point Round interrupt is taken.

Table 63-30. Embedded Floating-point Round interrupt—register settings

Register	Setting description					
SRR0	Set to the effective address of the instruction following the excepting EFPU instruction.					
SRR1	Set to the contents of the MSR at the time of the interrupt					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	—	IS	0
	CE	—	FE0	0	DS	0
	EE	0	DE	—	PMM	0
	PR	0			RI	—
ESR	SPV, VLEMI. All other bits cleared.					
MCSR	Unchanged					
DEAR	Unchanged					
Vector	IVPR[0:23] || 0xB0					

63.8.5.13 System Reset interrupt

The System Reset exception is a non-maskable, asynchronous exception signaled to the processor through the assertion of system-defined signals.

A System Reset may be initiated by either a software reset or during power-on reset.

When a reset request occurs, the processor branches to the system reset exception vector without attempting to reach a recoverable state. If reset occurs during normal operation, all operations cease and the machine state is lost.

The following table lists register settings when a System Reset interrupt is taken.

Table 63-31. System Reset interrupt—register settings

Register	Setting description					
CSRR0	Undefined					
CSRR1	Undefined					
MSR	SPV	0	FP	0	FE1	0
	WE	0	ME	0	IS	0

Table continues on the next page...

Table 63-31. System Reset interrupt—register settings (continued)

Register	Setting description					
	CE	0	FE0	0	DS	0
	EE	0	DE	0	PMM	0
	PR	0			RI	0
ESR	Cleared					
DEAR	Undefined					
Vector	[**p_rstbase[0:29]**] || 2'b00					

63.9 End-to-End ECC support

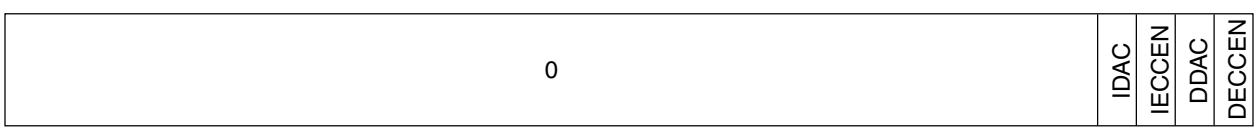
This section describes end-to-end error detection and correction capability (e2eECC) enhancements to address additional diagnostic capabilities for the next generation of embedded designs.

63.9.1 End-to-End ECC control and configuration

e2eECC is controlled by a set of privileged device control registers (DCRs) accessed using the **mfdcr** and **mtdcr** instructions. These registers and the operation of various features are described in the following subsections.

63.9.1.1 End-to-End ECC Control Register 0 (E2ECTL0)

The End-to-End ECC Control Register 0 (E2ECTL0) controls operation of the e2eECC logic.



DCR - 510; Read/Write; Reset - 0x5 || p_e2e_rstcfg[0:3]; Privileged

Figure 63-18. e2eECC Control Register 0 (E2ECTL0)

Table 63-32. E2ECTL0 field descriptions

Bits	Name	Description
0:27	—	Reserved
28	IDAC	Instruction Disable Address Checking

Table continues on the next page...

Table 63-32. E2ECTL0 field descriptions (continued)

Bits	Name	Description
		This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Instruction AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[0]** input.
29	IECCEN	Instruction ECC Enable Field 0 End-to-End ECC is disabled for the Instruction interface. No checking of instruction fetch data is performed. 1 End-to-End ECC is enabled for performing error detection and correction on the Instruction interface. Checking of instruction fetch data is performed with correction of single-bit data errors. Detection of any address errors or uncorrectable multi-bit data errors cause a machine check to be generated if the instruction fetch information is attempted to be used for instruction decoding and execution. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[1]** input.
30	DDAC	Data Disable Address Checking This bit controls usage of the address portion of the ECC H matrix. When use is disabled, the address portion is not used for checkbit/syndrome generation for Data AHB accesses. 0 Use of access address is enabled in checkbit and syndrome generation. 1 Use of access address is disabled in checkbit and syndrome generation. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[2]** input.
31	DECCEN	Data ECC Enable Field 0 End-to-End ECC is disabled for the data interface. No checking of read data is performed. Writes will still generate the proper check bit values to be supplied to external storage devices. 1 End-to-End ECC is enabled for performing error detection and correction on the data interface. Checking of read data is performed with correction of single-bit data errors. Detection of any address errors, or uncorrectable multi-bit data errors cause a machine check to be generated. Writes generate the proper check bit values to be supplied to external storage devices. **Note:** This field is updated on **p_reset_b** based on the value of the **p_e2e_rstcfg[3]** input.

e2eECC is enabled by setting the IECCEN or DECCEN bit of the E2ECTL0 DCR to a 1. On reset, e2eECC operation may be disabled from detecting or correcting errors based on reset configuration inputs, and no exceptions will be signaled for nonzero calculated syndrome values. This allows for the proper initialization of stored checkbits in any volatile storage by the CPU without causing error conditions due to uninitialized storage. Following the initialization, software may enable e2eECC operation by writing the appropriate values to the E2ECTL0[IECCEN] or E2ECTL0[DECCEN] control bits.

The E2ECTL0[IDAC] or E2ECTL0[DDAC] control bits can be used to remove the address checking component of the ECC logic for the Instruction AHB and Data AHB when address inclusion is not desired.

63.9.1.2 End-to-End ECC error generation by software

Error generation provides a way to test error recovery and portions of the error detection/correction logic by intentionally injecting parity errors into the driven checkbit information on the external bus during write operations. No provision is made to generate internal errors on read data due to timing issues; instead, software may intentionally generate errors in the checkbit values to emulate data, address, or checkbit error on external bus write cycles, and read back the written data to cause an error to be detected.

The End-to-End ECC Error Control/Status Register (E2EECSR0) controls operation of the e2eECC error injection logic. It allows for generation of a single error injection operation on the next CPU data write to the external bus only, or for a series of error injection write operations to the external bus, using the WRC and INVC control fields. The WRCHKBIT/CHKINVT field controls which of the **p_hwchkbit[7:0]** outputs are affected.

Control is provided to allow for either inverting the value(s) of one or more of the checkbit outputs **p_hwchkbit[7:0]** in hardware on one or more external write accesses, or for software to directly supply checkbit values to be used for the external write access(es).

The End-to-End ECC Error Control/Status Register (E2EECSR0) also supports access to raw checkbit values via the RCHKBIT status field. This field is updated with checkbit values received on each CPU data read operation to the external bus (but not on cache hits).

Note that Nexus 3 accesses do not cause error injection or the capture of read checkbits, regardless of the settings of the E2EECSR0 register.

The following figure shows the layout of E2EECSR0.

0	RCHKBIT	0	WRC	INVC	0	WCHKBIT/CHKINVT
0 1 2 3	4 5 6 7 8 9 10 11	12 13 14 15	16 17	18 19	20 21 22 23	24 25 26 27 28 29 30 31

DCR - 511; Read/Write; Reset - 0x0; Privileged

Figure 63-19. e2eECC Error Control/Status Register 0 (E2EECSR0)

Table 63-33. E2EECSR0 field descriptions

Bits	Name	Description
0:3	—	Reserved
4:11	RCHKBIT	Read Checkbits This field provides the raw checkbits received on the last CPU data access to external memory via the Data BIU. This field corresponds bit-wise to **p_hrchkbit[7:0]** for external reads. Software may use this information to determine the stored checkbits of external memories by performing CPU load operations and then accessing this field prior to the next load operation (assuming interrupts are masked).
12:15	—	Reserved
16:17	WRC	Write Control 00 No write checkbit substitution is performed by this control bit 01 Checkbit substitution is performed for the next CPU external write access (only) by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. Software must clear this field before rewriting to 01 in order to generate a subsequent checkbit substitution operation. 10 Checkbit substitution is performed for each CPU external write access while this field remains set to 10 by driving **p_hwchkbit[7:0]** with the values in the WCHKBIT control field. This field must be cleared by software to discontinue further checkbit substitution. 11 Reserved Note: Checkbit substitution has priority over Error injection
18:19	INVC	Invert Control 00 No error injection is performed by this control bit 01 Error injection is performed for the next CPU external write access (only) by modifying **p_hwchkbit[7:0]** based on CHKINVT. Software must clear this field before rewriting to 01 in order to generate a subsequent error injection operation. 10 Error injection is performed for each CPU external write access while this field remains set to 10 by modifying **p_hwchkbit[7:0]** based on CHKINVT. This field must be cleared by software to discontinue further error generation. 11 Reserved Note: Checkbit substitution has priority over Error injection
20:23	—	Reserved
24:31	WCHKBIT/ CHKINVT	Write Checkbits / Checkbit Invert Mask This field provides the checkbit substitution values when performing checkbit substitution via the WRC control field, and controls which checkbits are inverted when performing error injection via the INVC control field. This field corresponds bit-wise to **p_hwchkbit[7:0]**. For Checkbit inversion operations via error injection: 0 Checkbit is driven normally 1 Checkbit is inverted before being driven out on **p_hwchkbit[7:0]** when error injection occurs.

Chapter 64
Z2 Core Debug Support

64.1 Overview

Internal debug support in the e200z210n3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware based debugger which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of *Power ISA 2.06*. e200z210n3 supports a subset of these defined facilities. In addition to the facilities defined in *Power ISA 2.06*, e200z210n3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

64.1.1 Software Debug Facilities

e200z210n3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 ($DBCR0_{IDM}$). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When $DBCR0_{IDM}$ is cleared (and $EDBCR0_{EDM}$ is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when $DBCR0_{IDM}$ is cleared (or is overridden by $EDBCR0_{EDM}$ being set and EDBRAC0 indicating no resource is "owned" by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

64.1.1.1 *Power ISA 2.06* Compatibility

The e200z210n3 core implements a subset of the *Power ISA 2.06* internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

64.1.2 Additional Debug Facilities

In addition to the debug functionality defined in *Power ISA 2.06*, e200z210n3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger. (See Sharing Debug Resources by Software/Hardware.) A data trace port is also provided.

64.1.2.1 Data Trace Port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z210n3 core. This port provides information corresponding to each read or write access completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (i.e. the

normal setting. See the "Data Trace Control Register (DTC)" section in the Core (e200z210n3) Nexus 3 Module chapter. The data trace port will report aligned accesses and misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0.

64.1.3 Hardware Debug Facilities

The e200z210n3 core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard IEEE 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the e200z210n3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug Mode enable bit in External Debug Control register 0 ($EDBCR0_{EDM}$), which is also aliased to $DBCR0_{EDM}$. Setting $EDBCR0_{EDM}$ overrides the Internal Debug Mode enable bit $DBCR0_{IDM}$ unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the "hardware-owned" debug facilities. In addition, since resources are "owned" by the hardware debugger, inconsistent values may be present if software attempts to read "hardware-owned" debug-related resources.

When hardware debug is enabled by setting $EDBCR0_{EDM}$=1, the control registers and resources described in Debug registers are reserved for use by the external debugger. The same events described in Software Debug Events and Exceptions are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. $EDBCR0_{EDM}$, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

[SCG_NX_DEBUG_ACCESS]Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.[end]

64.1.4 Sharing Debug Resources by Software/Hardware

Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When $EDBCR0_{EDM}$ is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources which set DBSR bits when $DBCR0_{IDM}$=1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources which set EDBSR0 bits when $EDBCR0_{EDM}$=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

64.1.4.1 Simultaneous Hardware and Software Debug Event Handing

Since it is possible that a "hardware-owned" resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented which guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

The following figure shows the e200z210n3 debug resources.

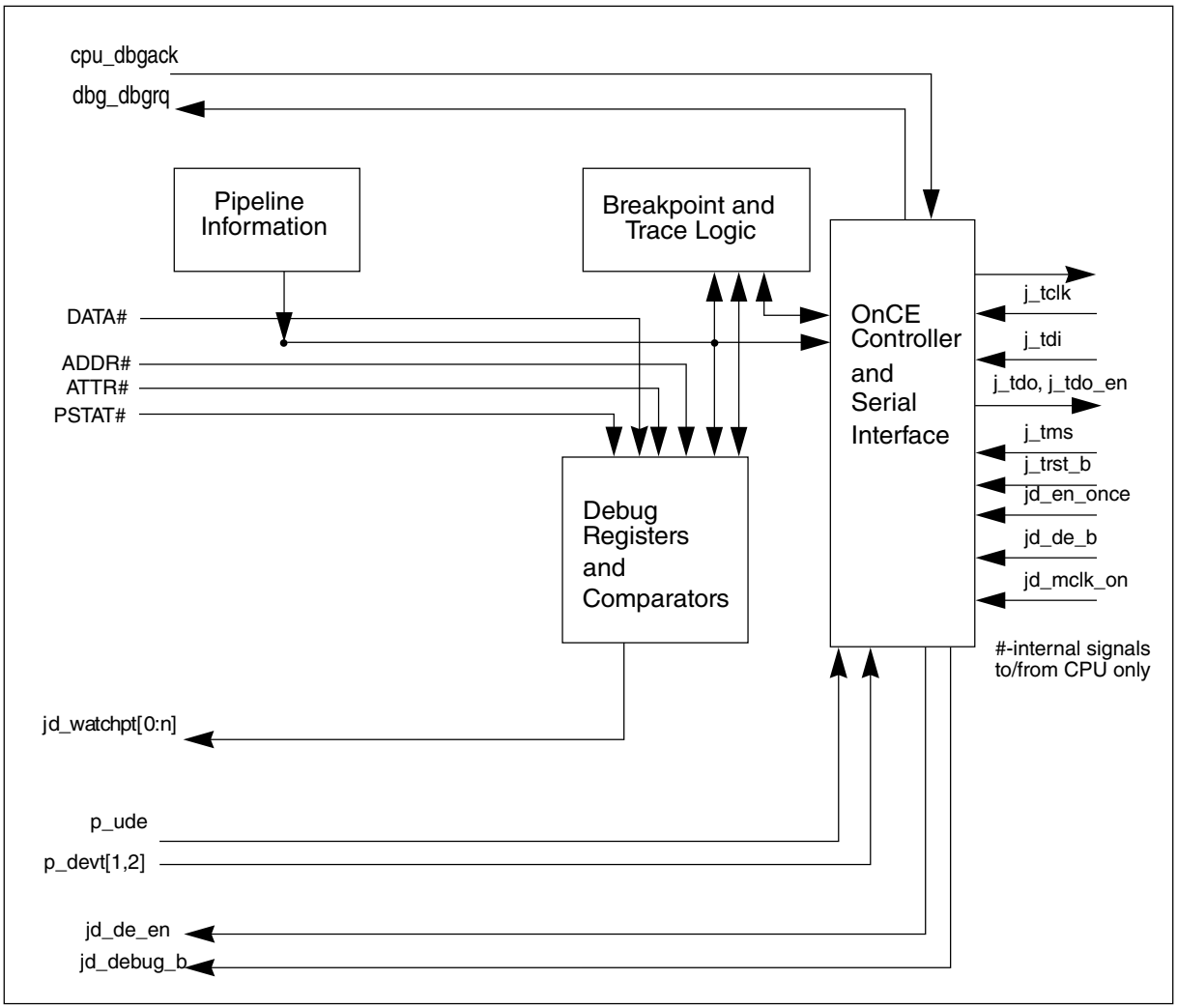


Figure 64-1. e200z210n3 Debug Resources

64.2 Software Debug Events and Exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}$=1) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in EDBRAC0). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers (DBCR0–8). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource which is "owned" by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE}, a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when MSR_{DE}=0 and $DBCR0_{IDM}$=1. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE}. A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while MSR_{DE}=0, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction which enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** which causes both MSR_{DE} and $DBCR0_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by *Power ISA 2.06*:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, e200z210n3 defines additional debug events:

- The External debug events DEVT1 and DEVT2 which are described in External debug event.
- The Performance Monitor Interrupt event PMI which is described in Performance Monitor Interrupt debug event.

The e200z210n3 debug configuration supports most of these event types. Unsupported *Power ISA 2.06* defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported

A brief description of each of the event types follows.

64.2.1 Instruction Address Compare Event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

64.2.2 Data Address Compare Event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load or store operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note

In contrast to the *Power ISA 2.06* definition, Data Address Compare events on e200z210n3 do not prevent the load or store class instruction from completing. If a load or store class

instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting $DBSR_{IDE}$) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $DBSR_{DAC_OFST}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the $DBSR_{IDE}$ bit will be set. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note

DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

Note

DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

Note

DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.

Note

Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.

64.2.2.1 Data Address Compare Event Status Updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $DBSR_{DAC_OFST}$ field when the DVC DAC status is recorded.

Table 64-1 outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by DSRR0.

Table 64-1. DAC events and Resultant Updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.

Table continues on the next page...

Table 64-1. DAC events and Resultant Updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Ldst instruction,no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

Table continues on the next page...

Table 64-1. DAC events and Resultant Updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DVC DACx	DVC DACy,Normal Ldst	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	Any instruction(no exception)	DSI, with or without DAC,Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	Any instruction(no exception)	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

64.2.3 Linked Instruction Address Compare and Data Address Compare Events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1 may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can

occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note

Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

64.2.4 Trap Debug Event

A Trap debug event (TRAP) occurs if Trap debug events are enabled ($DBCR0_{TRAP}$=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the $DBSR_{TRAP}$ bit is set to 1 to record the debug exception.

64.2.5 Branch Taken Debug Event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled ($DBCR0_{BRT}$=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus $DBSR_{IDE}$ can not be set by a Branch Taken debug event. When a Branch Taken debug event is recognized, the $DBSR_{BRT}$ bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

64.2.6 Instruction Complete Debug Event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled ($DBCR0_{ICMP}$=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The ***se_sc*** instruction does not fall into the category of an instruction whose execution is

suppressed, since the instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR_{DE}=0 at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

Note

Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to '1' while $DBCR0_{ICMP}$=1, nor by the execution of an instruction that sets $DBCR0_{ICMP}$ to '1' while MSR_{DE}=1.

64.2.7 Interrupt Taken Debug Event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}$=1) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using SRR0/1) cause an Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the non-critical interrupt handler.

64.2.8 Critical Interrupt Taken Debug Event

A Critical Interrupt Taken debug event (CIRPT) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}$=1) and a critical interrupt occurs. Only critical class interrupts (an interrupt using CSRR0/1) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the critical interrupt handler.

64.2.9 Return Debug Event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}$=1) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfi** (i.e. before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

64.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}$=1) and an attempt is made to execute an **se_rfci** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfci** (i.e. before the MSR is updated by the **se_rfci**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfci**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfci** instruction.

64.2.11 External debug event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}$=1 or $DBCR0_{DEVT2}$=1), and the respective **p_devt1** or **p_devt2** input signal transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to '1' to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

64.2.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to '1' to record the debug exception. This debug event is an asynchronous event.

64.2.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}$=1), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to '1' to record the debug exception. This debug event is an asynchronous event.

64.3 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register EDBRAC0, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in EDBRAC0 corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not "owned" by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not "owned" by software will not cause modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not "owned" by software is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the $DBCR0_{EDM}$ bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

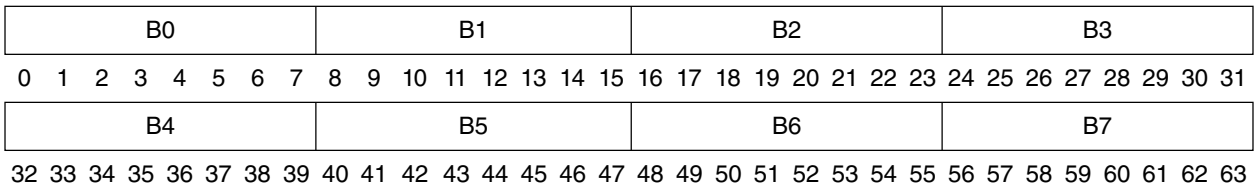
64.3.1 Debug Address and Value Registers

[SCG_NX_DEBUG_REGS]Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in the following figure) corresponds to the byte data value transferred to/from memory byte offset 0, 8, …, and the least significant byte of the register (labeled B7 in the following figure) corresponds to byte offset 7, F, … . When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison, they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z210n3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfspr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.[end]



SPR - 601 (DVC1U), 318 (DVC1), 602 (DVC2U) 319 (DVC2); Read/Write; Reset - Unaffected

Figure 64-2. DVC1, DVC2 registers

64.3.2 Debug Control and Status registers

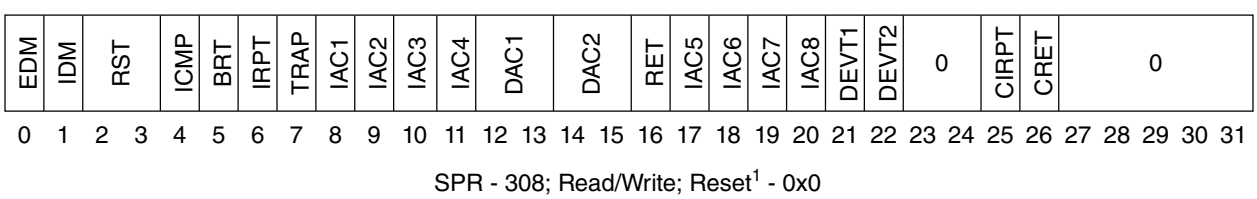
Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

e200z210n3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

64.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z210n3 adds some implementation specific bits to this register, as seen in the following figure.



SPR - 308; Read/Write; Reset[1] - 0x0

Figure 64-3. Debug Control Register 0 (DBCR0) register

Note

[1] $DBCR0_{EDM}$ is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if $DBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $DBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b**, and only software-owned resources indicated by EDBRAC0 and the $DBCR0_{RST}$ field will be reset by **p_reset_b**. The $DBCR0_{RST}$ field will always be reset by **p_reset_b** regardless of the value of $DBCR0_{EDM}$.

The following provides bit definitions for Debug Control Register 0.

Table 64-2. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset p_reset_b. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCR0–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0. Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to '1' before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to '1', all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If MSR_{DE}=1, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.
2:3	RST	Reset Control 00 No function 01 **p_dbrstc[1]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 **p_dbrstc[0]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved

Table continues on the next page...

Table 64-2. DBCR0 field descriptions (continued)

Bit	Name	Description
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled 1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled
10	IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11	IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13	DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15	DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses 11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16	RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled
17	IAC5	Instruction Address Compare 5 Debug Event Enable

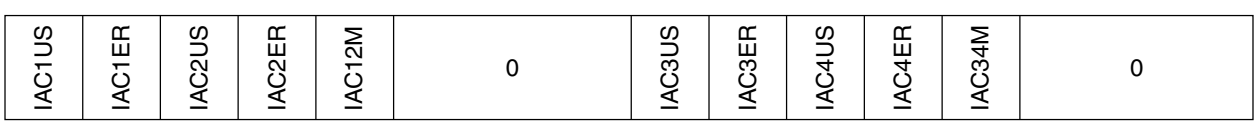
Table continues on the next page...

Table 64-2. DBCR0 field descriptions (continued)

Bit	Name	Description
		0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1- IAC8 debug events are enabled
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

64.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in the following figure.



SPR - 309; Read/Write; Reset[1] - 0x0

Figure 64-4. DBCR1 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 1.

Table 64-3. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode 00 IAC1 debug events not affected by MSR_{PR} 01 Reserved 10 IAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC1 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode 00 IAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode 00 IAC2 debug events not affected by MSR_{PR} 01 Reserved 10 IAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode 00 IAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC12M	Instruction Address Compare 1/2 Mode 00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2. 01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used. 10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.

Table continues on the next page...

Table 64-3. DBCR1 field descriptions (continued)

Bit	Name	Description
		11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
10:15	—	Reserved
16:17	IAC3US	Instruction Address Compare 3 User/Supervisor Mode 00 IAC3 debug events not affected by MSR_{PR} 01 Reserved 10 IAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC3 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC3ER	Instruction Address Compare 3 Effective/Real Mode 00 IAC3 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.
26:31	—	Reserved

64.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.The DBCR2 register is shown in the following figure.

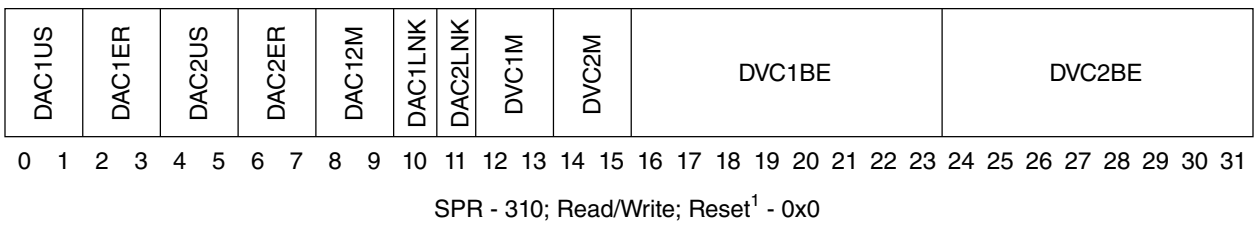


Figure 64-5. DBCR2 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 2.

Table 64-4. DBCR2 Bit Definitions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR_{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR_{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address

Table continues on the next page...

Table 64-4. DBCR2 Bit Definitions (continued)

Bit	Name	Description
		01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM.
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies Exact Address Compare since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR
12:13	DVC1M	Data Value Compare 1 Mode When $DBCR4_{DVC1C}$=0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC1C}$=1: 00 Reserved

Table continues on the next page...

Table 64-4. DBCR2 Bit Definitions (continued)

Bit	Name	Description
		01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Note: Inactive byte lanes of the memory access are automatically masked.
14:15	DVC2M	Data Value Compare 2 Mode When $DBCR4_{DVC2C}$=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC2C}$=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Inactive byte lanes of the memory access are automatically masked.
16:23	DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.

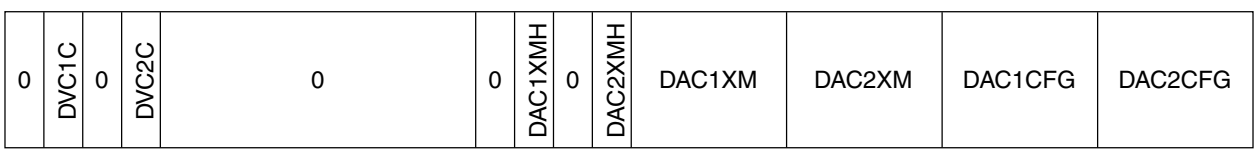
Table continues on the next page...

Table 64-4. DBCR2 Bit Definitions (continued)

Bit	Name	Description
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

64.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in the following figure.



SPR - 563; Read/Write; Reset[1] - 0x0

Figure 64-6. DBCR4 Register

Note

[1] DBCR4 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 4.

Table 64-5. DBCR4 field description

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC1 operation. 1 Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition 0 Normal DVC2 operation. 1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field. 0 DAC1XM masks 0–15 low-order address bits 1 DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. . DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH || DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH || DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}$=00 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}$=00

Table continues on the next page...

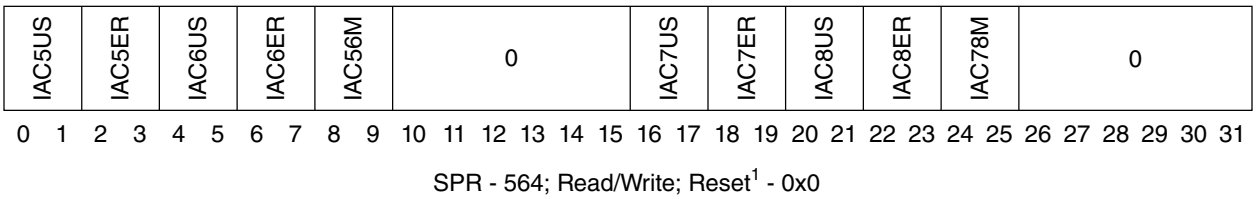
Table 64-5. DBCR4 field description (continued)

Bit	Name	Description
		0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}$=00 0011 Reserved 01xx Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). **NOTE:** unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}$=0; when $DBCR0_{DAC1}$!=00, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}$=00 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}$=00 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}$=00 0011 Reserved 01xx Reserved 1xxx Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. **NOTE:** Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}$=00. When $DBCR0_{DAC2}$!=00, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.

64.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in the following figure.

.



SPR - 564; Read/Write; Reset[1] - 0x0

Figure 64-7. DBCR5 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 5.

Table 64-6. DBCR5 field descriptions

Bit(s)	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR_{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR_{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC56M	Instruction Address Compare 5/6 Mode

Table continues on the next page...

Table 64-6. DBCR5 field descriptions (continued)

Bit(s)	Name	Description
		00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR_{PR} 01 Reserved 10 IAC7 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR_{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8.

Table continues on the next page...

Table 64-6. DBCR5 field descriptions (continued)

Bit(s)	Name	Description
		01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

64.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in the following figure.

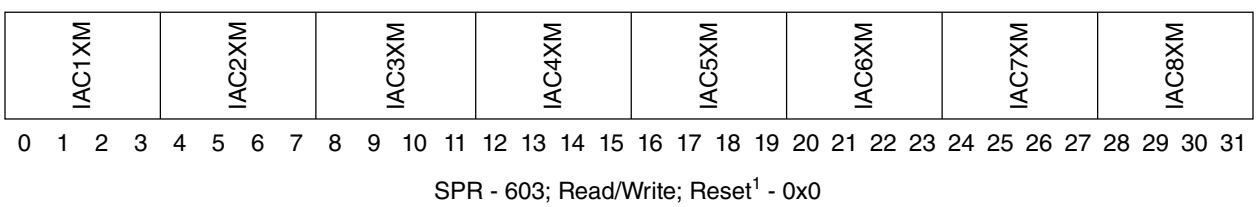


Figure 64-8. DBCR6 Register

Note

[1] DBCR6 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 6.

Table 64-7. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M]=00

Table continues on the next page...

Table 64-7. DBCR6 field descriptions (continued)

Bit	Name	Description
		0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Controll. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8. 0000 No additional masking when DBCR5[IAC78M]=00

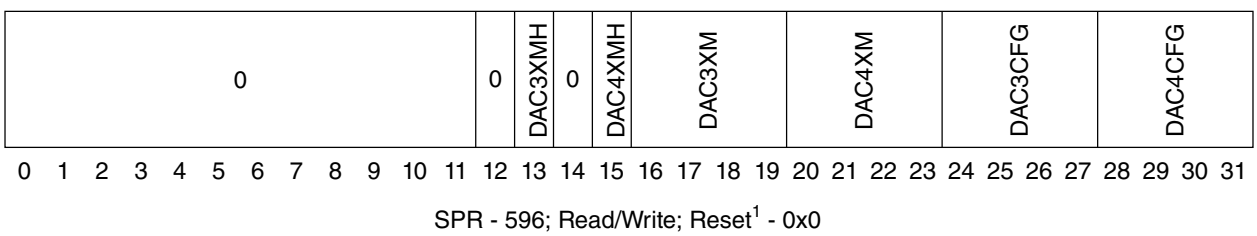
Table continues on the next page...

Table 64-7. DBCR6 field descriptions (continued)

Bit	Name	Description
		0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved

64.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in the following figure.



SPR - 596; Read/Write; Reset[1] - 0x0

Figure 64-9. DBCR7 Register

Note

[1] DBCR7 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 7.

Table 64-8. DBCR7 Field Descriptions

Bit	Name	Description
0:12	—	Reserved
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High .DAC3XMH extends the range of the DAC3XM field

Table continues on the next page...

Table 64-8. DBCR7 Field Descriptions (continued)

Bit	Name	Description
		0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. . DAC4XMH extends the range of the DAC4XM field. 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4. Value of DAC3XMH || DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3. Value of DAC4XMH || DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match

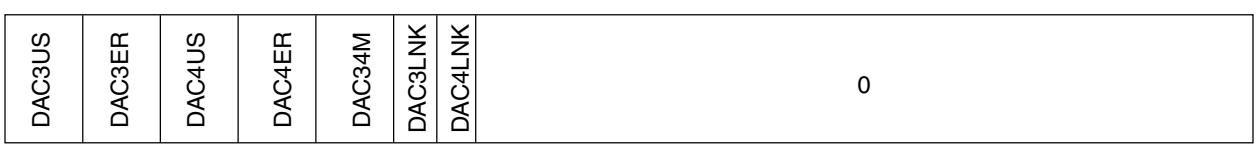
Table continues on the next page...

Table 64-8. DBCR7 Field Descriptions (continued)

Bit	Name	Description
		occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001 - 1111 Reserved
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1xxx Reserved

64.3.2.8 Debug Control Register 8 (DBCR8)

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in the following figure.



SPR - 597; Read/Write; Reset[1] - 0x0

Figure 64-10. DBCR8 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 8.

Table 64-9. DBCR8 Bit Definitions

Bit(s)	Name	Description
0:1	DAC3US	Data Address Compare 3 User/Supervisor Mode 00 - DAC3 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC3 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	DAC3ER	Data Address Compare 3 Effective/Real Mode 00 - DAC3 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=0 11 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC4US	Data Address Compare 4 User/Supervisor Mode. 00 - DAC4 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC4 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC4ER	Data Address Compare 4 Effective/Real Mode 00 - DAC4 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=0 11 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC34M	Data Address Compare 3/4 Mode 00 - Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4.

Table continues on the next page...

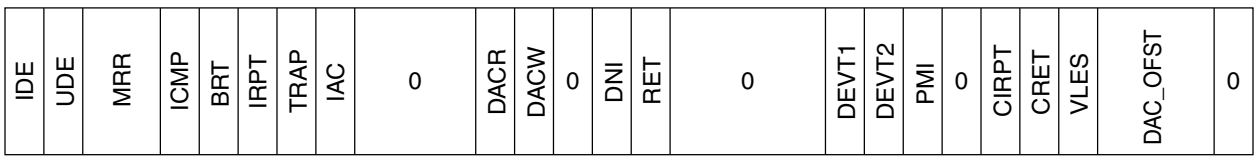
Table 64-9. DBCR8 Bit Definitions (continued)

Bit(s)	Name	Description
		01 - Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 - Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 - Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.
10	DAC3LNK	Data Address Compare 3 Linked 0 - No effect 1 - DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 - No effect 1 - DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

64.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr***DBSR,RS*. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. Debug Status bits are set by Debug events only while Internal Debug Mode is enabled ($DBCR0_{IDM}$=1). When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1, $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource

sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in the following figure.



SPR - 304; Read/Clear; Reset - 0x1000_0000

Figure 64-11. DBSR Register

The following table provides bit definitions for the Debug Status Register.

Table 64-10. DBSR Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if MSR_{DE}=0 and $DBCR0_{IDM}$=1 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store which is terminated with error.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 - No reset occurred since these bits were last cleared by software 01 - A hard reset occurred since these bits were last cleared by software 10 - Reserved 11 - Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a Trap Taken debug event occurred.
8	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event

Table continues on the next page...

Table 64-10. DBSR Bit Definitions (continued)

Bit(s)	Name	Description
		Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UDI}$=1
24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a Power ISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

64.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC). DDEAR is update by hardware with the effective address of the load or store operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in the following table.

Data Effective Address																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 600; Read/Write; Reset - unaffected

Figure 64-12. DDEAR Register

64.3.3 External Debug Resource Allocation Control Register (EDBRAC0)

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}$=1. EDBRAC0 settings are ignored when $EDBCR0_{EDM}$=0.

Hardware-owned resources which generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources which generate debug events if $DBCR0_{IDM}$=1 update DBSR, causing debug interrupts to occur if MSR_{DE}=1. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug Mode is enabled. When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event which is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug Mode is enabled ($EDBCR0_{EDM}$=1). When $EDBCR0_{EDM}$=1, a set bit in EDBSR0 by an event which is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}$=1, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in DBCR0–8 are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all

registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in the following figure.

0	IDM	RST	UDE	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC34	DAC2	0	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	PMI	0	CIRPT	CRET	DNI	DQM	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29 30 31

SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b,** reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

Figure 64-13. EDBRAC0 Register

Table 64-10 provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 64-11. EDBRAC0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control 0 - Internal Debug mode may not be enabled by software. $DBCR0_{IDM}$ is owned exclusively by hardware. **mtspr** DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 - Internal Debug mode may be enabled by software. $DBCR0_{IDM}$ is owned by software. $DBCR0_{IDM}$ is software readable/writable. When $EDBRAC0_{IDM}$=1, software writes to hardware-owned bits in DBCR0–8 via **mtspr** are ignored.
2	RST	Reset Field Control 0 - $DBCR0_{RST}$ owned exclusively by hardware debug. No **mtspr** access by software to $DBCR0_{RST}$ field. 1 - $DBCR0_{RST}$ accessible by software debug. $DBCR0_{RST}$ is software readable/writable.
3	UDE	Unconditional Debug Event 0 - Event owned by hardware debug. 1 - Event owned by software debug.
4	ICMP	Instruction Complete Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{ICMP}$ field. 1 - Event owned by software debug. $DBCR0_{ICMP}$ is software readable/writable.
5	BRT	Branch Taken Debug Event

Table continues on the next page...

Table 64-11. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{BRT}$ field. 1 - Event owned by software debug. $DBCR0_{BRT}$ is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{IRPT}$ field. 1 - Event owned by software debug. $DBCR0_{IRPT}$ is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{TRAP}$ field. 1 - Event owned by software debug. $DBCR0_{TRA}$P is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC1 control and status fields. 1 - Event owned by software debug. IAC1 control fields are software readable/writable.
9	IAC2	Instruction Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC2 control and status fields. 1 - Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC3 control and status fields. 1 - Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC4 control and status fields. 1 - Event owned by software debug. IAC4 control fields are software readable/writable.
12	DAC1	Data Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC1 control and status fields. 1 - Event owned by software debug. DAC1 control fields are software readable/ writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 - Events owned by hardware debug. No **mtspr** access by software to DAC3 and DAC4 control and status fields. 1 - Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.
14	DAC2	Data Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC2 control and status fields. 1 - Event owned by software debug. DAC2 control fields are software readable/ writable.
15	—	Reserved
16	RET	Return Debug Event

Table continues on the next page...

Table 64-11. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{RET}$ field. 1 - Event owned by software debug. $DBCR0_{RET}$ is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC5 control and status fields. 1 - Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC6 control and status fields. 1 - Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC7 control and status fields. 1 - Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC8 control and status fields. 1 - Event owned by software debug. IAC8 control are software readable/writable.
21	DEVT1	External Debug Event Input 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT1}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT1}$ is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT2}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT2}$ is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to the PMRs. Performance monitor interrupts set $EDBSR0_{PMI}$ regardless of the setting of $PMGC0_{UDI}$ 1 - Event owned by software debug. PMRs are software readable/writable. **Note:** this bit is reset to '1'.
24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CIRPT}$ field. 1 - Event owned by software debug. $DBCR0_{CIRPT}$ is software readable/writable.
26	CRET	Critical Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CRET}$ field. 1 - Event owned by software debug. $DBCR0_{CRET}$ is software readable/writable.
27	DNI	DNI Instruction Debug Control

Table continues on the next page...

Table 64-11. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		0 - DNI resource owned by hardware debug. Execution of an **e_dni** or **se_dni** instruction results in entry into debug mode. 1 - DNI resource owned by software debug. Execution of an **e_dni** or **se_dni** instruction results in either a debug interrupt ($DBCR0_{IDM}$=1 and MSR_{DE}=1) or a nop ($DBCR0_{IDM}$=0 or MSR_{DE}=0). **Note:** DNI events are not blocked during a debug session
28	DQM	Data Acquisition Messaging Registers 0 - $DEVENT_{DQTAG}$ and DDAM register are exclusively owned by hardware debug. No **mtspr** access by software to $DEVENTD_{QTAG}$ field or DDAM register. Attempted access by software is ignored. 1 - $DEVENT_{DQTAG}$ and DDAM register are owned by software. Software has read/write access to $DEVENT_{DQTAG}$ field and DDAM register.
29:31	—	Reserved

Table 64-12 shows which resources are controlled by EDBRAC0 settings.

Table 64-12. EDBRAC0 Resource Control

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$		$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	**Software Accessible via mtspr,affected by p_reset_b**
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	All debug registers
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IDM}$
1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{RST}$
1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{UDE}$
1	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{ICMP}$
1	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{BRT}$
1	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IRPT}$
1	1	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{TRAP}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC1, $DBCR0_{IAC1}$, $DBCR1_{IAC1US,IAC1ER}$, $DBCR6_{IAC1XM}$
1	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC2, $DBCR0_{IAC2}$, $DBCR1_{IAC2US,IAC2ER}$, $DBCR6_{IAC2XM}$

Table continues on the next page...

Table 64-12. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$		$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
1	1	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC12M}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC3, $DBCR0_{IAC3}$, $DBCR1_{IAC3US,IAC3ER}$, $DBCR6_{IAC3XM}$
1	1	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC4, $DBCR0_{IAC4}$, $DBCR1_{IAC4US,IAC4ER}$, $DBCR6_{IAC4XM}$
1	1	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC34M}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC5, $DBCR0_{IAC5}$, $DBCR5_{IAC5US,IAC5ER}$, $DBCR6_{IAC5XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC6, $DBCR0_{IAC6}$, $DBCR5_{IAC6US,IAC6ER}$, $DBCR6_{IAC6XM}$
1	1	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC56M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC7, $DBCR0_{IAC7}$, $DBCR5_{IAC7US,IAC7ER}$, $DBCR6_{IAC7XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	IAC8, $DBCR0_{IAC8}$, $DBCR5_{IAC8US,IAC8ER}$, $DBCR6_{IAC8XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC78M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	DAC1, DVC1, DVC1U

Table continues on the next page...

Table 64-12. EDBRAC0 Resource Control (continued)

EDBCR0$_{EDM}$	EDBRAC0$_{IDM}$	EDBRAC0$_{RST}$	EDBRAC0$_{UDE}$	EDBRAC0$_{ICMP}$	EDBRAC0$_{BRT}$	EDBRAC0$_{IRPT}$	EDBRAC0$_{TRAP}$	EDBRAC0$_{IAC1}$	EDBRAC0$_{IAC2}$	EDBRAC0$_{IAC3}$	EDBRAC0$_{IAC4}$	EDBRAC0$_{IAC5}$	EDBRAC0$_{IAC6}$	EDBRAC0$_{IAC7}$	EDBRAC0$_{IAC8}$	EDBRAC0$_{DAC1}$	EDBRAC0$_{DAC34}$	EDBRAC0$_{DAC2}$	EDBRAC0$_{RET}$	EDBRAC0$_{DEVT1}$	EDBRAC0$_{DEVT2}$	EDBRAC0$_{PMI}$		EDBRAC0$_{CIRT}$	EDBRAC0$_{CRET}$	EDBRAC0$_{DNI}$	EDBRAC0$_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
																												DBCR0$_{DAC1}$, DBCR2$_{DAC1US}$,DAC1ER, DBCR2$_{DVC1M}$,DVC1BE DBCR4$_{DVC1C}$,DAC1XM
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DAC3, DAC4 DBCR7$_{DAC\{3,4\}}$,DAC\{3,4\}CFG, DAC\{3,4\}XM, DAC\{3,4\}XMH DBCR8$_{DAC\{3,4\}US}$, DAC\{3,4\}ER, DAC\{3,4\}M
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	DAC2, DVC2, DVC2U DBCR0$_{DAC2}$, DBCR2$_{DAC2US}$,DAC2ER, DBCR2$_{DVC2M}$,DVC2BE DBCR4$_{DVC2C}$,DAC2XM
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	-	-	-	-	-	-	-	-	-	DBCR2$_{DAC12M}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	DBCR2$_{DAC1LNK}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	DBCR2$_{DAC2LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DBCR8$_{DAC3LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DBCR8$_{DAC4LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	DBCR0$_{RET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	DBCR0$_{DEVT1}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	DBCR0$_{DEVT2}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	All PMRs
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	DBCR0$_{CIRPT}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	DBCR0$_{CRET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	1	-	
1	-[1]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	DEVENT$_{DQTAG}$, DDAM

1. Note: IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8 and the PMRs are reset by assertion of **p_reset_b** when $EDBCR0_{EDM}$=1. Only software-owned bits or fields as shown in Table 64-12 are affected in this case, except that $DBCR0_{RST}$ and $DBSR_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of $EDBCR0_{EDM}$ or EDBRAC0.

64.3.4 Debug Event Select Register (DEVENT)

The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering. See the SoC User's Manual for more information on SoC use cases.

The $DEVENT_{DEVNT}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. See the "Data Acquisition ID Tag Field" section in the Core (e200z210n3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in the following figure.

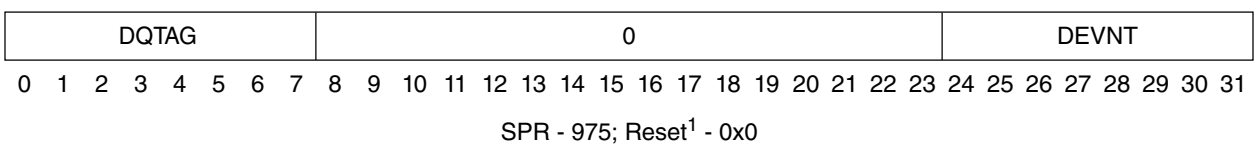


Figure 64-14. DEVENT Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

The following table provides bit definitions for the Debug Event Register.

Table 64-13. DEVENT Bit Definitions

Bit(s)	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.
24:31	DEVNT	Debug Event Signals 00000000 - No signal is asserted xxxxxxx1 - **p_devnt_out[0]** and **jd_watchpt[12]** are asserted for one clock xxxxxx1x - **p_devnt_out[1]** and **jd_watchpt[13]** are asserted for one clock xxxxx1xx - **p_devnt_out[2]** and **jd_watchpt[20]** are asserted for one clock xxxx1xxx - **p_devnt_out[3]** and **jd_watchpt[21]** are asserted for one clock xxx1xxxx - **p_devnt_out[4]** is asserted for one clock xx1xxxxx - **p_devnt_out[5]** is asserted for one clock x1xxxxxx - **p_devnt_out[6]** is asserted for one clock 1xxxxxxx - **p_devnt_out[7]** is asserted for one clock

64.3.5 Debug Data Acquisition Message Register (DDAM)

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this register. See the "Data Acquisition Messaging" section in the Core (e200z210n3) Nexus 3 Module chapter for details.

The DDAM register is shown in the following figure.

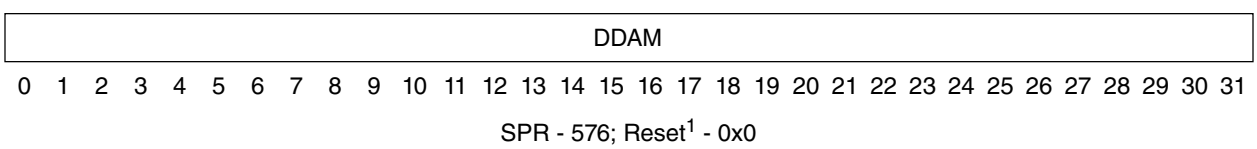


Figure 64-15. DDAM Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for the Debug Data Acquisition Message Register.

Table 64-14. DDAM Bit Definitions

Bit(s)	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

64.4 Using Debug Resources for Stack Limit Checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DAC resources configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0 pointing to the address of the load or store access which generated the stack overflow/underflow. If DAC resources configured to perform stack limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in Table 64-5. Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the $DBCR4_{DAC1CFG}$ field to '1000' or the $DBCR8_{DAC3CFG}$ field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '10', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '01', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, i.e. after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the $EDBSR0_{IDE}$ status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is

aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow, and for user mode accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in Table 64-5 and Table 64-8 in Debug Control and Status registers. Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

64.5 External Debug Support

External debug support is supplied through the OnCE controller serial interface which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the $EDBCR0_{EDM}$/$DBCR0_{EDM}$ bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When $EDBCR0_{EDM}$ is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug Mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug Mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of Software Debug Events and Exceptions refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note

On the initial setting of $EDBCR0_{EDM}$ to '1', other bits in DBCR0 will remain unchanged. After $EDBCR0_{EDM}$ has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information which could cause unintended operation.

Note

It is intended for the CPU to remain in external debug mode ($EDBCR0_{EDM}$=1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.

Note

$EDBCR0_{EDM}$ operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if $EDBCR0_{EDM}$ was previously set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all *hardware* debug events are blocked. Watchpoints are not blocked.

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the $EDBCR0_{EDM}$ bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is

performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

64.5.1 External Debug Registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug Mode.

64.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software, However, the state of $EDBCR0_{EDM}$ is reflected as a read-only bit in $DBCR0_{EDM}$ to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in the following figure.

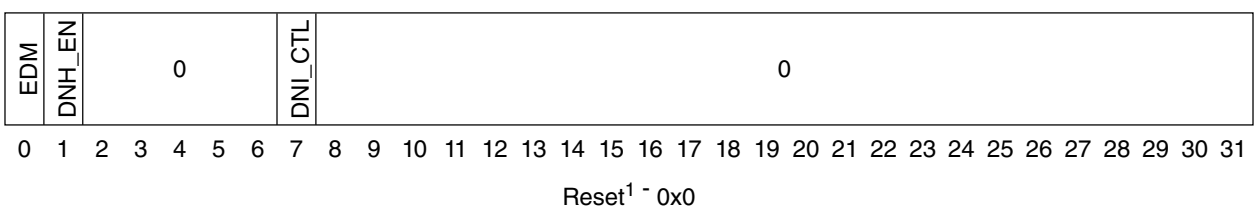


Figure 64-16. EDBCR0 Register

Note

[1] EDBCRO is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b.**

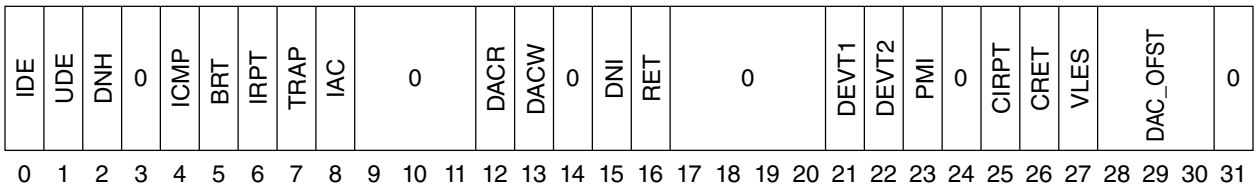
The following table provides bit definitions for External Debug Control Register 0.

Table 64-15. EDBCR0 Bit Definitions

Bit(s)	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0 0 - External debug mode disabled. Internal debug events not mapped into external debug events. 1 - External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1-8, DAC1-4, DVC1-2[U]} unless permitted by settings in EDBRAC0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events which remain active across a processor reset.
1	DNH_EN	**dnh** Instruction Enable 0 - execution of **e_dnh** and **se_dnh** instructions cause illegal instruction exceptions to occur. 1 - execution of **e_dnh** and **se_dnh** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	---	Reserved
7	DNI_CTL	**dni** Instruction Control 0 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is cared, and when MSRDE=0, execution of **e_dni** and **se_dni** instructions are nop'ed and no entry into debug mode occurs. 1 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is don't-cared, and execution of **e_dni** and **se_dni** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR_{DE}. Note that this control bit is only used when the dni resource is owned by hardware via control in $EDBRAC0_{DNI}$, and thus also only when EDM=1
8:31	---	Reserved

64.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in the following figure.



Read/Write; Reset[1] - 0x0000_0000

Figure 64-17. EDBSR0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register 0.

Table 64-16. EDBSR0 Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if $EDBCR0_{EDM}$=1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store which is terminated with error. This bit will not be set for imprecise debug events which are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event

Table continues on the next page...

Table 64-16. EDBSR0 Bit Definitions (continued)

Bit(s)	Name	Description
		Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $EDBRAC0_{PMI}$=0 regardless of the setting of $PMGC0_{UDI}$
24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a Power ISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by $EDBCR0_{DNH_EN}$. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $EDBSR0_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

64.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A '1' stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events which are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes

without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in the following figure.

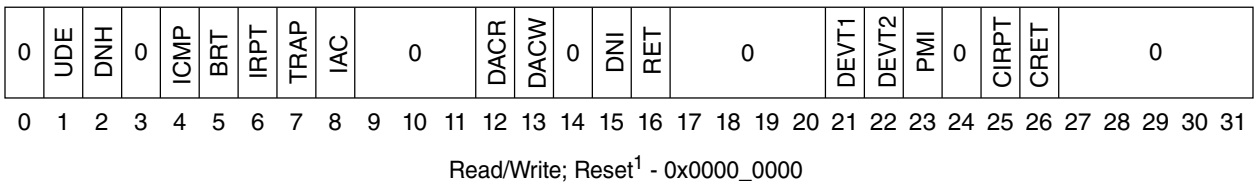


Figure 64-18. EDBSRMSK0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=O.

The following table provides bit definitions for External Debug Status Register Mask 0.

Table 64-17. EDBSRMSK0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{UDE}$
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by $EDBSR0_{DNH}$
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{ICMP}$
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{BRT}$
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IRPT}$
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{TRAP}$
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IAC}$
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACR}$
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACW}$
14	—	Reserved
15	DNI	DNI Debug Event

Table continues on the next page...

Table 64-17. EDBSRMSK0 Bit Definitions (continued)

Bit(s)	Name	Description
		Set to 1 to mask debug mode entry by $EDBSR0_{DNI}$
16	RET	Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{RET}$
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT1}$
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT2}$
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{PMI}$
24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CIRPT}$
26	CRET	Critical Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CRET}$
22:31	—	Reserved

64.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events. EDDEAR is update by hardware with the effective address of the load or store operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of $EDBSR0_{DAC\{R,W\}}$ bits which are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the $EDBSR0_{DAC\{R,W\}}$ bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in the following figure.

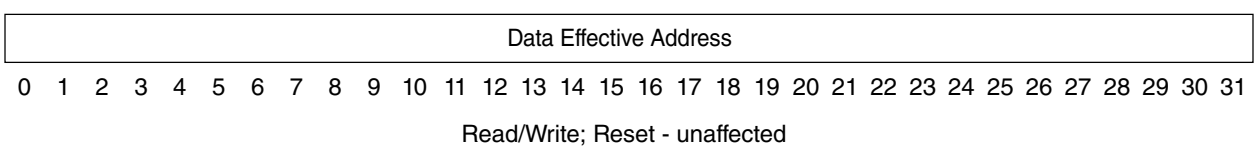


Figure 64-19. EDDEAR Register

64.5.2 OnCE Introduction

The e200z210n3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z210n3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

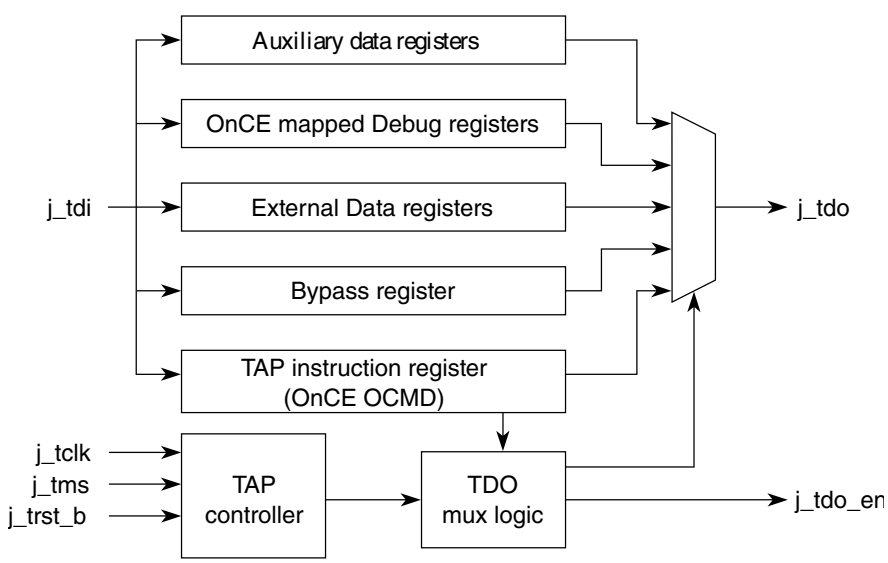
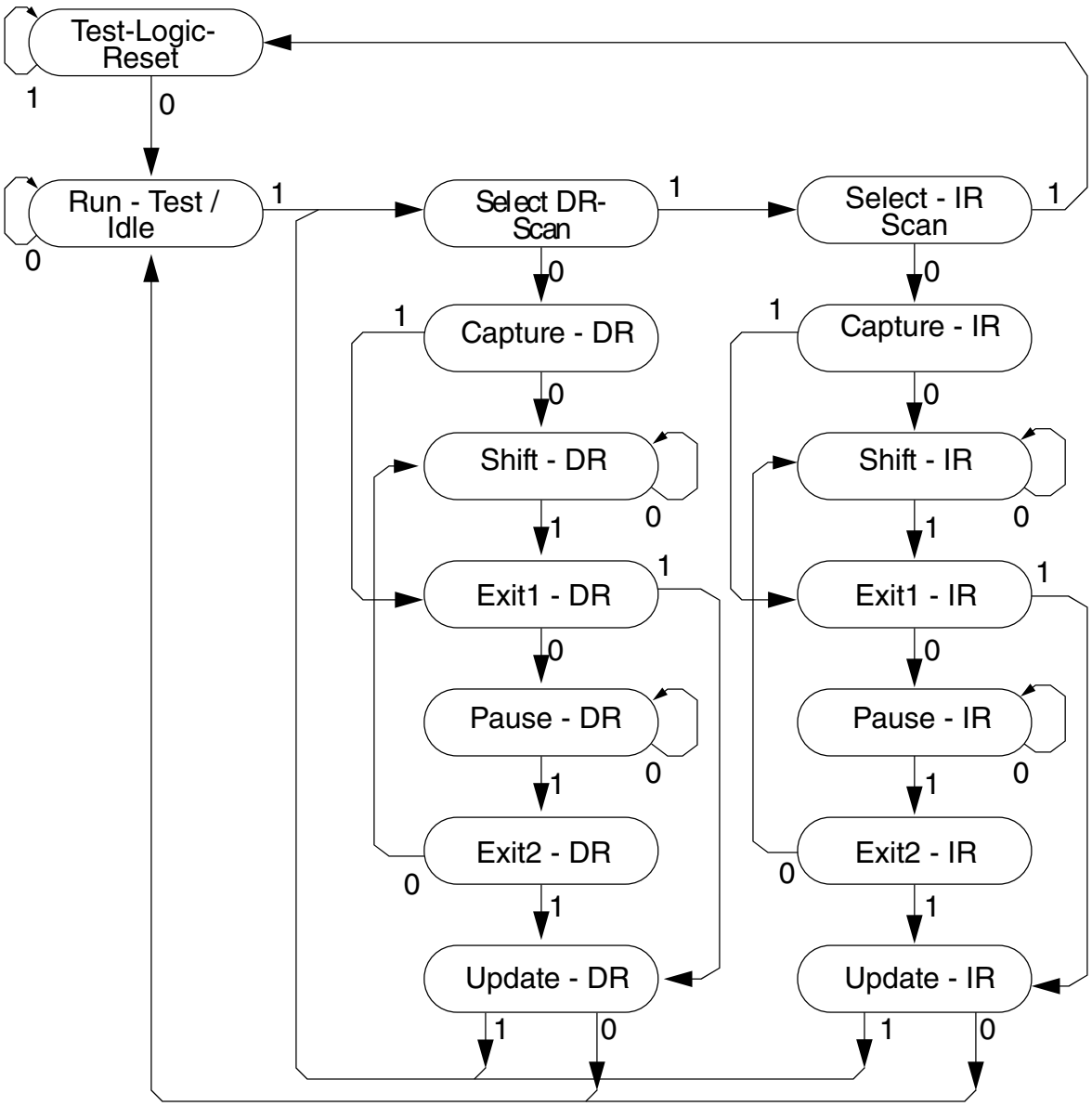


Figure 64-20. OnCE TAP Controller and Registers

The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.



Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), OnCE Status Register) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will indicate the CPU is in the debug state. Instructions may the be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (See OnCE Command Register (OCMD)). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

64.5.3 JTAG/OnCE Pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

The following table details the primary JTAG/OnCE interface signals.

Table 64-18. JTAG/OnCE Primary Interface Signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi[0:31]	I	JTAG test data or parallel input
j_tdo	O	Test data out to master controller or pad
j_tdo_en[1]	O	Enables TDO output buffer
j_parallel_IR_en	I	Enables parallel loading of IR during Shift_IR state
j_parallel_DR_en	I	Enables parallel loading of DR during Shift_DR state

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z210n3) Core Complex Overview.

64.5.4 OnCE Internal Interface Signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

64.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

64.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

64.5.4.3 CPU Address, Attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

64.5.4.4 CPU Data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

64.5.5 OnCE Interface Signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks which may need information pertaining to debug operation.

64.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE commands are executed by the OnCE unit, and all other commands default to a "Bypass" command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regsel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/ Idle, or Update_DR TAP states. A new value will take affect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

64.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z210n3 interface) provides a fast means of entering the Debug Mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug Mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug Mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug Mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug Mode. Once Debug Mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems which do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

64.5.5.3 OnCE Debug Output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control which are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either "No Register Selected" or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. See OnCE Command Register (OCMD) for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

64.5.5.4 CPU Clock On Input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU's **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z210n3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

64.5.5.5 Watchpoint Events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, and the Performance Monitor control register settings. Refer to Table 64-24 for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and

may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

64.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates "no register selected". This signal will assert regardless of whether the CPU is currently in debug mode. It may be monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with "no register selected", and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs which are not it debug mode will ignore the command.

64.5.6 OnCE Controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. The following figure is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in or parallel loading commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR and Update-DR states.

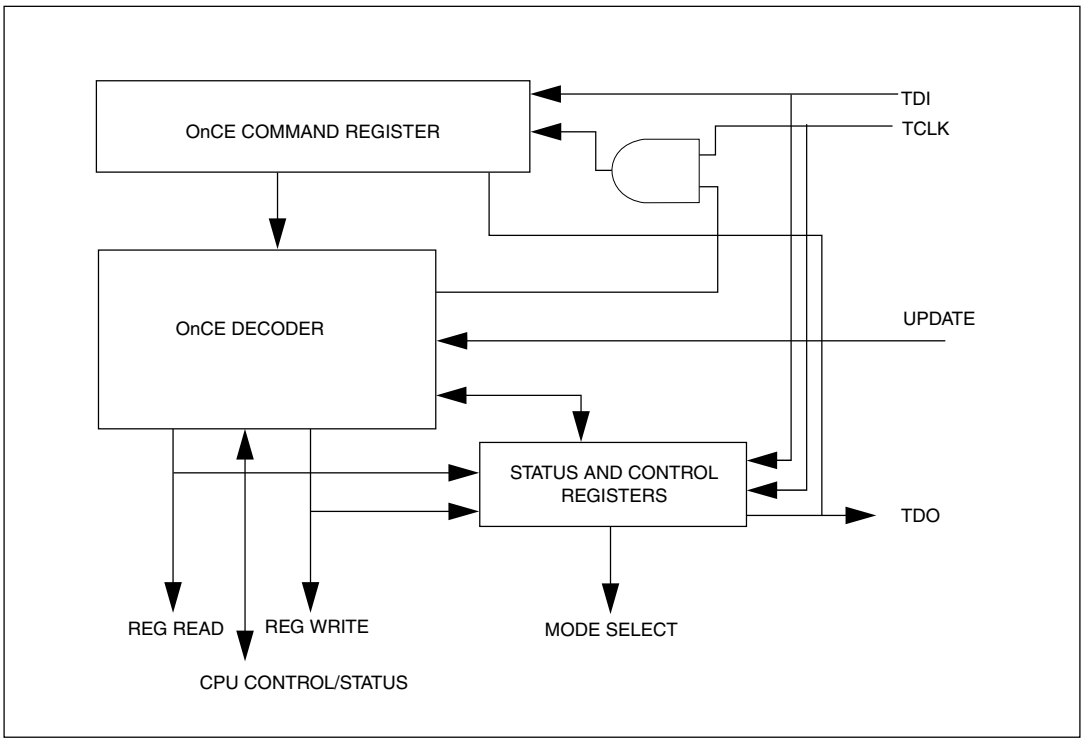


Figure 64-21. OnCE Controller and Serial Interface

64.5.6.1 OnCE Status Register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo**[31] output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first. Parallel output of the IR status (OSR) is available via the j_tdo[22:31] outputs during the first cycle of the Shift_IR state(s), with j_tdo[22] corresponding to the most-significant bit of the OSR (MCLK).

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

Figure 64-22. OnCE Status Register

The following table provides bit definitions for the Once Status Register.

Table 64-19. OnCE Status Register Bit Definitions

Bit(s)	Name	Description
0	MCLK	MCLK **m_clk** Status Bit

Table continues on the next page...

Table 64-19. OnCE Status Register Bit Definitions (continued)

Bit(s)	Name	Description
		0 - Inactive state 1 - Active state This status bit reflects the logic level on the **jd_mclk_on** input signal after capture by **j_tclk**.
1	ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including External, Critical, machine check, Storage, Alignment, Program, etc.) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
3	RESET	RESET Mode This bit reflects the inverted logic level on the CPU **p_reset_b** input after capture by **j_tclk**.
4	HALT	HALT Mode This bit reflects the logic level on the CPU **p_halted** output after capture by **j_tclk**.
5	STOP	STOP Mode This bit reflects the logic level on the CPU **p_stopped** output after capture by **j_tclk**.
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU **p_waiting** output after capture by **j_tclk**.
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

64.5.6.2 OnCE Command Register (OCMD)

The OnCE Command Register (OCMD) is a 10-bit shift register that receives its serial or parallel data from the TDI pins and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in Figure 64-23. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In

addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.



Reset - 10'b1000000010 on assertion of **j_trst_b** or **m_por** , or while in the Test_Logic_Reset state

Figure 64-23. OnCE Command Register

Table 64-20 provides bit definitions for the Once Command Register.

Table 64-20. OnCE Command Register Bit Definitions

Bit(s)	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. **Note:** The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on **j_tdo[31]** during the first 32 clocks of Shift-DR, or on the j_tdo[0:31] outputs during the first clock of Shift-DR. 0 - Write the data associated with the command into the register specified by RS[0:6] 1 - Read the data contained in the register specified by RS[0:6]
1	GO	Go Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction which resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the GO bit is ignored.The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking. 0 - Inactive (no action taken) 1 - Execute instruction in IR
2	EX	Exit Command Bit 0 - Remain in debug mode 1 - Leave debug mode

Table continues on the next page...

Table 64-20. OnCE Command Register Bit Definitions (continued)

Bit(s)	Name	Description
		If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. Note that if the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and $EDBCR0_{EDM}$=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode *without* execution of an instruction, even though the EX bit was set. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking.
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. Table 64-21 indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

[SCG_NX_ONCE_REGS]Table 64-21 indicates the OnCE register addresses.[end]

Table 64-21. OnCE Register Addressing

RS[0:6]	Register Selected
000 0000	Reserved
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)
010 0110	Data Value Compare 1 (DVC1) - *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) - *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)

Table continues on the next page...

Table 64-21. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Reserved for Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Reserved for Cache Debug Access Control Register (CDACNTL)
111 1011	Reserved for Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3-Access (<<<put in cross reference to Nexus 3 Module>>>)
111 1101	LSRL Select (see Test Specification)
111 1110	Enable_OnCE[1]
111 1111	Bypass

1. Causes assertion of the j_en_once_regsel output.

The Once Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

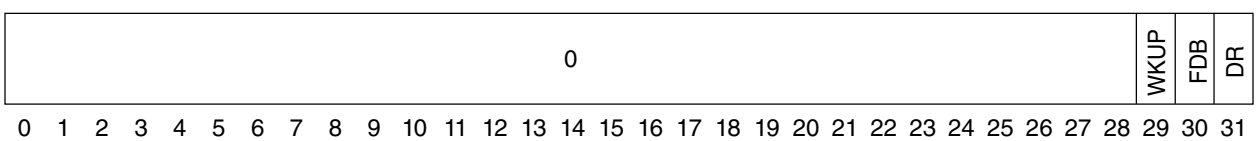
Single stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (interrupts, machine checks, bus error conditions, etc.) and may prevent the desired instruction from being successfully executed. The OSR_{ERR} bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

Additionally, the $EDBCR0_{EDM}$ bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of EDBCR0EDM. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

64.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z210n3 core into debug mode and to enable / disable sections of the OnCE control logic. The control bits are read/write. These bits are only effective while OnCE is enabled (**jd_en_once** asserted). The OCR is shown in the following figure.



Reset - 0xo000_0000 on **m_por, j_trst_b,** or entering Test_logic_Reset state

Figure 64-24. OnCE Control Register

The following table provides bit definitions for the OnCE Control Register.

Table 64-22. OnCE Control Register Bit Definitions

Bit(s)	Name	Description
0: 28	—	Reserved
29	WKUP	Wakeup Request Bit (WKUP)

Table continues on the next page...

Table 64-22. OnCE Control Register Bit Definitions (continued)

Bit(s)	Name	Description
		This control bit may be used to force the **p_wakeup** output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the **m_clk** input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.
30	FDB	Force Breakpoint Debug Mode Bit (FDB) This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the '**bkpt**' pseudo- instruction will cause the processor to enter debug mode, as if the **jd_de_b** input had been asserted. This bit is qualified with $EDBCR0_{EDM}$, which must be set for FDB to take effect. Note that this bit has no effect on **e_dnh** or **se_dnh** instruction operation.
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter the Debug Mode. The CPU will indicate that Debug Mode has been entered via the data scanned out in the shift-IR state. 0 - No Debug Mode request 1 - Unconditional Debug Mode request When the DR bit is set the processor will enter Debug mode at the next instruction boundary.

64.5.7 Access to Debug Resources

Resources contained in the OnCE Module which do not require the processor core to be halted for access may be accessed while the e200z210n3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z210n3 core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the e200z210n3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the OCR_{WKUP} control bit.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (DBSR, etc.) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

The following table provides a list of access requirements for OnCE registers.

Table 64-23. OnCE Register Access Requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 $_{EDM}$ = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
Enable_OnCE	N	N	N	N	—	
Bypass	N	N	N	N	N	
CPUSCR	Y	Y	Y	Y	Y	
DAC1–4	Y	Y	Y	N	*[1]	
DBCR0	Y	Y	Y	N	*[1]	*EDBCR0$_{EDM}$ access only requires **jd_en_once** asserted
DBCR1–8	Y	Y	Y	N	*[1]	
DEVENT	Y	Y	Y	N	*[1]	
EDBRAC0 (DBERC0)	Y	N	Y	N	*[1]	
DBSR	Y	Y	Y	N[2]	*[1]	
EDBCR0	Y	N	N	N	N	
EDBSR0	Y	N	Y	N	N	
EDBSRMSK0	Y	N	N	N	N	
IAC1–8	Y	Y	Y	N	*[1]	
JTAG ID	N	N	—	N	—	Read-only
OCR	Y	N	N	N	N	
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while **jd_en_once** is asserted
Nexus3-Access	Y	N	N	N	N	
PMR-Access	Y	N	N	N	N	
PMR Registers	Y	Y	Y	N	N	
External GPRs	Y	N	N	N	N	
LSRL Select	Y	N	?	?	?	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.

2. Reads of these registers while the CPU is running may not give data that is self-consistent due to synchronization across clock domains.

64.5.8 Methods of Entering Debug Mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug Mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input See OnCE Enable (jd_en_once).

64.5.8.1 External Debug Request During RESET

[SCG_NX_EXT_DEBUG_REQ]Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b,** and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z210n3 core to enter Debug Mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the e200z210n3 core does not execute an instruction before entering Debug Mode, although the first instruction to be executed will be fetched prior to entering Debug Mode in order to properly initialize the CPUSCR.[end]

64.5.8.2 Debug Request During RESET

[SCG_NX_DEBUG_DUR_RESET]Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.[end]

64.5.8.3 Debug Request During Normal Activity

[SCG_NX_DEBUG_NORMAL1]Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.[end]

64.5.8.4 Debug Request During Waiting, Halted or Stopped State

[SCG_NX_DEBUG_DUR_WAIT_HALT_STOP]Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting, p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled along with other status bits from the **j_tdo** outputs during the Shift_IR TAP controller state. This status may be used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.[end]

64.5.8.5 Software Request During Normal Activity

[SCG_NX_DEBUG_NORMAL2]Upon executing a '**bkpt**' pseudo-instruction (for e200z210n3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and $EDBCR0_{EDM}$=1, the CPU enters the debug mode after the instruction following the '**bkpt**' pseudo-instruction has entered the instruction register.[end]

64.5.8.6 Debug Notify Halt Instructions

[SCG_NX_DEBUG_NOTIFY_HALT]The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by $EDBCR0_{DNH_EN}$, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is ***not*** conditioned on $EDBCR0_{EDM}$, allowing for debug of software debug handlers as well as other software. For e200z210n3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.[end]

64.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

[SCG_NX_CPU_STATUS] A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. The following figure shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.[end]

NOTE

Note that due to the length of this register, parallel JTAG access capability is not provided.

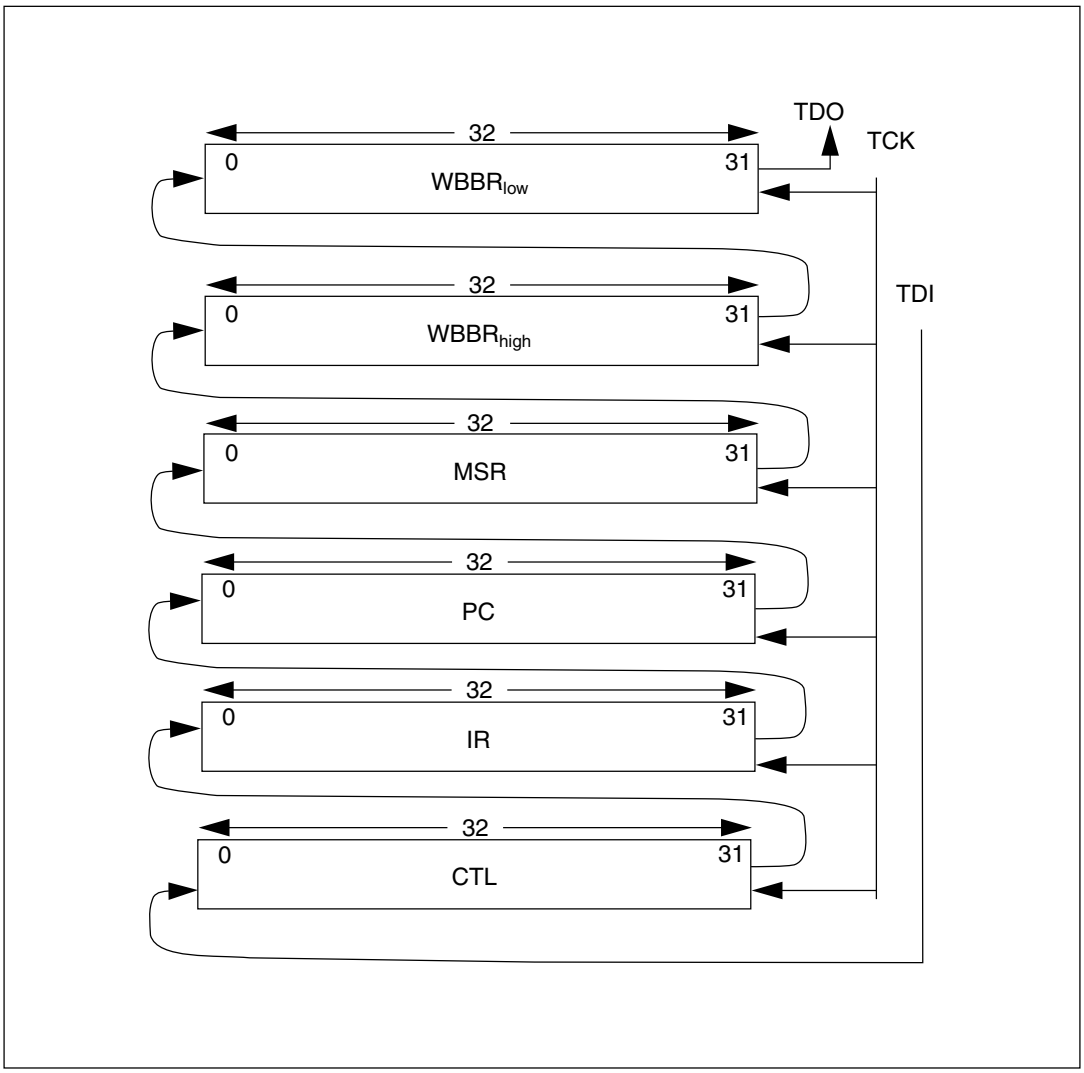


Figure 64-25. CPU Scan Chain Register (CPUSCR)

64.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug

control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

64.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the operations performed during the debug session and should normally be restored by the external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

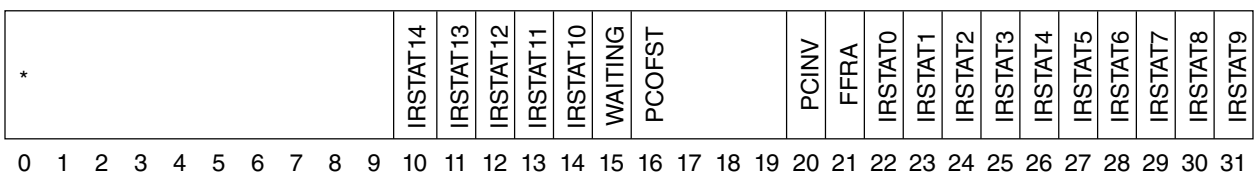


Figure 64-26. Control State Register (CTL)

WAITING — WAITING State Status

This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit.

0- CPU was not in the waiting state when debug mode was entered

1- CPU was in the waiting state when debug mode was entered

PCOFST — PC Offset Field

This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead of the original IR value, other wise the original value of IR should be restored. (But see PCINV which overrides this field)

0000 - No correction required.

0001 - Subtract 0x04 from PC.

0010 - Subtract 0x08 from PC.

0011 - Subtract 0x0C from PC.

0100 - Subtract 0x10 from PC.

0101 - Subtract 0x14 from PC.

all other encodings are reserved

* — Internal State Bits

Table continues on the next page...

These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. See OnCE Debug Output (jd_debug_b), which is not part of the normal program execution flow, these bits should be set to a 0.

PCINV — PC and IR Invalid Status Bit

This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered.

0= No error condition exists.

1= Error condition exists. PC and IR are corrupted.

FFRA— Feed Forward RA Operand Bit

This control bit causes the content of the $WBBR_{lo}$ to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR.

This allows the debug firmware to update processor registers — initialize the $WBBR_{lo}$ with the desired value, set the FFRA bit, and execute a ori Rx,Rx,0 instruction to the desired register.

Note: not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent.

0= No action.

1= Content of $WBBR_{lo}$ used as operand value

IRStat0 — IR Status Bit 0

This control bit indicates a TEA status for the IR.

0= No TEA occurred on the fetch of this instruction.

1= TEA occurred on the fetch of this instruction.

IRStat1 — IR Status Bit 1

This control bit is reserved.

IRStat2 — IR Status Bit 2

This control bit indicates an Instruction Address Compare 1 event status for the IR.

0= No Instruction Address Compare event 1 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 1 occurred on the fetch of this instruction.

IRStat3 — IR Status Bit 3

This control bit indicates an Instruction Address Compare 2 event status for the IR.

0= No Instruction Address Compare 2 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 2 event occurred on the fetch of this instruction.

IRStat4 — IR Status Bit 4

This control bit indicates an Instruction Address Compare 3 event status for the IR.

0= No Instruction Address Compare event 3 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 3 occurred on the fetch of this instruction.

IRStat5 — IR Status Bit 5

This control bit indicates an Instruction Address Compare 4 event status for the IR.

Table continues on the next page...

0= No Instruction Address Compare 4 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 4 event occurred on the fetch of this instruction.

IRStat6 — IR Status Bit 6

This control bit indicates a Parity Error status for the IR.

0= No Parity Error occurred on the fetch of this instruction.

1= Parity Error occurred on the fetch of this instruction .

IRStat7 — IR Status Bit 7

This control bit indicates a Precise External Termination Error status for the IR.

0= No Precise External Termination Error occurred on the fetch of this instruction.

1= A Precise External Termination Error occurred on the fetch of this instruction.

IRStat8 — IR Status Bit 8

This control bit indicates the Power ISA VLE status for the IR.

0= IR contains a BookE instruction. (unused encoding)

1= IR contains a Power ISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit.

- this bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).

IRStat9 — IR Status Bit 9

This control bit is reserved.

IRStat10 — IR Status Bit 10

This control bit indicates an Instruction Address Compare 5 event status for the IR.

0= No Instruction Address Compare 5 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 5 event occurred on the fetch of this instruction.

IRStat11 — IR Status Bit 11

This control bit indicates an Instruction Address Compare 6 event status for the IR.

0= No Instruction Address Compare 6 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 6 event occurred on the fetch of this instruction.

IRStat12 — IR Status Bit 12

This control bit indicates an Instruction Address Compare 7 event status for the IR.

0= No Instruction Address Compare 7 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 7 event occurred on the fetch of this instruction.

IRStat13 — IR Status Bit 13

This control bit indicates an Instruction Address Compare 8 event status for the IR.

0= No Instruction Address Compare 8 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 8 event occurred on the fetch of this instruction.

IRStat14 — IR Status Bit 14

This control bit is reserved.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be

written with the FFRA bit set as appropriate, and all other bit set to '0', and the IR set to the value of the desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit which was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC+4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events which may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to '0's if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to '0' to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be re-entered prior to any further instruction execution.

64.5.9.3 Program Counter Register (PC)

The PC is a 32-bit register that stores the value of the program counter which was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

64.5.9.4 Write-Back Bus Register ($WBBR_{low}$, $WBBR_{high}$)

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. $WBBR_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. $WBBR_{high}$ holds the updated effective address calculated by a load with update instruction. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into $WBBR_{low}$. The contents of $WBBR_{low}$ can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the $WBBR_{low}$ to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to Control State Register (CTL) for more detail on the CTL_{FFRA} bit.)

$WBBR_{low}$ and $WBBR_{high}$ are generally undefined on instructions which do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

64.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used.This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

64.5.9.6 Exiting Debug Mode and Interrupt Blocking

When exiting debug mode with a Go+Exit, "asynchronous" interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will re-enter debug mode without executing an instruction

following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

64.5.10 Reserved Registers (Reserved)

The Reserved Registers are used to control various test control logic. These registers are not intended for customer use. To preclude device and/or system damage, these registers should not be accessed.

64.6 Watchpoint Support

e200z210n3 supports the generation and signalling of watchpoints when operating in internal debug mode ($DBCR0_{IDM}$=1) or in external debug mode ($EDBCR0_{EDM}$=1). Watchpoints are indicated with a dedicated set of interface signals. The **jd_watchpt[0:31]** output signals are used to indicate that a watchpoint has occurred. Certain watchpoints however (DEVENT-based, DTC-based, and Performance Monitor watchpoints) are not qualified with $EDBCR0_{EDM}$ or $DBCR0_{IDM}$.

Each debug address compare function (IAC1–8, DAC1–4), as well as other event types are capable of triggering a watchpoint output. The DBCRx control fields are used to configure watchpoints, regardless of whether events are enabled in DBCR0. Watchpoints may occur whenever an associated event would have been posted in the Debug Status Register if enabled. No explicit enable bits are provided for watchpoints; they are always enabled by definition, although the data address compare watchpoints may be controlled for read and write accesses via configuration fields in DBCR4, 7, and 9. During a debug session, debug events (other than PMI, DEVT1 and DEVT2) with a corresponding DBSR bit are blocked from asserting a watchpoint. The Performance Monitor, DEVNT-based and DTC-based watchpoints are not blocked during a debug session. If not desired, for address-based events the base address values for these events may be programmed to an unused system address. MSR_{DE} has no effect on watchpoint generation.

External logic may monitor the assertion of these signals for debugging or triggering purposes. Watchpoints are signaled in the clock cycle following the occurrence of the actual event. The Nexus3 module also monitors assertion of a portion of these output signals (**jd_watchpt[0:31]**) for various development control purposes. (See the "Watchpoint Trace Messaging" section in the Core (e200z210n3) Nexus 3 Module chapter.)

Note that these signals are **m_clk** domain signals, not **j_tclk** domain signals.

The following table shows the **jd_watchpt[0:31]** output signal assignments to various debug events.

Table 64-24. Watchpoint Output Signal Assignments

Signal Name	Type	Description
jd_watchpt[0]	IAC1	Instruction Address Compare 1 watchpoint Asserted whenever an IAC1 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[1]	IAC2	Instruction Address Compare 2 watchpoint Asserted whenever an IAC2 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[2]	IAC3	Instruction Address Compare 3 watchpoint Asserted whenever an IAC3 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[3]	IAC4	Instruction Address Compare 4 watchpoint Asserted whenever an IAC4 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[4]	DAC1	Data Address Compare 1 watchpoint Asserted whenever a DAC1 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC1CFG}$ settings
jd_watchpt[5]	DAC2	Data Address Compare 2 watchpoint Asserted whenever a DAC2 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC2CFG}$ settings
jd_watchpt[6]	DAC3	Data Address Compare 3 watchpoint Asserted whenever a DAC3 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC3CFG}$ settings
jd_watchpt[7]	DAC4	Data Address Compare 4 watchpoint Asserted whenever a DAC4 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC4CFG}$ settings
jd_watchpt[8]	IAC5	Instruction Address Compare 5 watchpoint Asserted whenever an IAC5 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[9]	IAC6	Instruction Address Compare 6 watchpoint Asserted whenever an IAC6 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[10]	DEVT1	Debug Event Input 1 watchpoint Asserted whenever a DEVT1 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[11]	DEVT2	Debug Event Input 2 watchpoint Asserted whenever a DEVT2 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[12]	DEVNT0	Debug Event Output 0 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[12]
jd_watchpt[13]	DEVNT1	Debug Event Output 1 watchpoint

Table continues on the next page...

Table 64-24. Watchpoint Output Signal Assignments (continued)

Signal Name	Type	Description
		Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[13]
jd_watchpt[14]	IAC7	Instruction Address Compare 7 watchpoint Asserted whenever an IAC7 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[15]	IAC8	Instruction Address Compare 8 watchpoint Asserted whenever an IAC8 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[16]	IRPT	Interrupt watchpoint Asserted whenever an IRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[17]	RET	Return watchpoint Asserted whenever a RET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[18]	CIRPT	Critical Interrupt watchpoint Asserted whenever a CIRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[19]	CRET	Critical Return watchpoint Asserted whenever a CRET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[20]	DEVNT2	Debug Event Output 2 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[20]
jd_watchpt[21]	DEVNT3	Debug Event Output 3 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[21]
jd_watchpt[22]	PMEVENT	Performance Monitor Event input watchpoint Asserted whenever **p_pm_event** transitions from a '0' to a '1' while **m_clk** is running
jd_watchpt[23]	PMC0	Performance Monitor Counter 0 watchpoint Asserted whenever PMC0 triggers an event based on $PMLCa0_{PMP}$
jd_watchpt[24]	PMC1	Performance Monitor Counter 1 watchpoint Asserted whenever PMC1 triggers an event based on $PMLCa1_{PMP}$
jd_watchpt[25]	PMC2	Performance Monitor Counter 2 watchpoint Asserted whenever PMC2 triggers an event based on $PMLCa2_{PMP}$
jd_watchpt[26]	PMC3	Performance Monitor Counter 3 watchpoint Asserted whenever PMC3 triggers an event based on $PMLCa3_{PMP}$
jd_watchpt[27]	—	Reserved
jd_watchpt[28]	TRAP	TRAP watchpoint Asserted whenever an TRAP debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[29]	DTC1	Data Trace Control Range 1 watchpoint

Table continues on the next page...

Table 64-24. Watchpoint Output Signal Assignments (continued)

Signal Name	Type	Description
		Asserted whenever an access meets the conditions for DTC Range 1
jd_watchpt[30]	DTC2	Data Trace Control Range 2 watchpoint Asserted whenever an access meets the conditions for DTC Range 2
jd_watchpt[31]	DTC3	Data Trace Control Range 3 watchpoint Asserted whenever an access meets the conditions for DTC Range 3

64.7 Basic Steps for Enabling, Using, and Exiting External Debug Mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug Mode and initializing Debug registers

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation.
- Select the OCR and write a value to it in which OCR_{DR}, OCR_{WKUP}, are set to '1'. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode.
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set.
- Clear the DBSR status bits.
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with "no-register selected", Go cleared, and determines that the CPU has re-entered the Debug state and that no ERR condition occurred.

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set.
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

To exit External Debug Mode

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR.
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the $EDBCR0_{EDM}$ bit.
- The debugger should write the DBSR to a cleared state.
- The debugger should re-write the DBCR0 with all bits including EDM cleared.
- The debugger should clear the OCR_{DR} bit.
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared.

Note

These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

Chapter 65
Z4 Core Debug Support

65.1 Overview

Internal debug support in the e200z4204n3 core allows for software and hardware debug by providing debug functions, such as instruction and data breakpoints and program trace modes. For software based debugging, debug facilities consisting of a set of software accessible debug registers and interrupt mechanisms are provided. These facilities are also available to a hardware based debugger which communicates using a modified IEEE 1149.1 Test Access Port (TAP) controller and pin interface. When hardware debug is enabled, the debug facilities controlled by hardware are protected from software modification.

Software debug facilities are defined as part of *PowerISA 2.06*. e200z4204n3 supports a subset of these defined facilities. In addition to the facilities defined in *PowerISA 2.06*, e200z4204n3 provides additional flexibility and functionality in the form of additional instruction breakpoints, linked instruction and data breakpoints, and extended range and value masking. These features are also available to a hardware-based debugger.

When not being used for debugging purposes, a portion of the debug facilities may be configured to provide a stack limit checking mechanism.

65.1.1 Software Debug Facilities

e200z4204n3 provides debug facilities to enable hardware and software debug functions, such as instruction and data breakpoints and program single stepping. The debug facilities consist of a set of debug control registers (DBCR0–2,4–8, EDBRAC0), a set of address compare registers (IAC1–8, DAC1–4), a set of 64-bit data value compare registers (DVC1, DVC2), a Debug Status Register (DBSR) for enabling and recording various kinds of debug events, a Debug Data Effective Address Register (DDEAR), and a special Debug interrupt type built into the interrupt mechanism.

The debug facilities also provide a mechanism for software-controlled processor reset, and debug mode entry. In addition, the Performance Monitor may be configured to utilize the debug interrupt type.

Software debug facilities are enabled by setting the internal debug mode bit in Debug Control register 0 ($DBCR0_{IDM}$). When internal debug mode is enabled, debug events can occur, and can be enabled to record exceptions in the Debug Status register (DBSR). If enabled by MSR_{DE}, these recorded exceptions cause Debug interrupts to occur. When $DBCR0_{IDM}$ is cleared (and $EDBCR0_{EDM}$ is cleared as well), no debug events occur, and no status flags are set in DBSR unless already set. In addition, when $DBCR0_{IDM}$ is cleared (or is overridden by $EDBCR0_{EDM}$ being set and EDBRAC0 indicating no resource is "owned" by software) no Debug interrupts will occur, regardless of the contents of DBSR. A software Debug interrupt handler may access all system resources and perform necessary functions appropriate for system debug.

65.1.1.1 *PowerISA 2.06* Compatibility

The e200z4204n3 core implements a subset of the *PowerISA 2.06* internal debug features. The following restrictions on functionality are present:

- Instruction address compares do not support compare on physical (real) addresses.
- Data address compares do not support compare on physical (real) addresses.

65.1.2 Additional Debug Facilities

In addition to the debug functionality defined in *PowerISA 2.06*, e200z4204n3 provides capability to link instruction and data breakpoints, provides additional instruction and data breakpoints, extended range and value masking, multiple watchpoint outputs, provides capability for the Performance Monitor to generate debug events, and allows for sharing of debug resources between software and a hardware debugger. (See Sharing Debug Resources by Software/Hardware.) A data trace port is also provided.

65.1.2.1 Data Trace Port

The data trace port interface is provided to assist in implementing extended debug watchpoint/breakpoint/trace capture capability with logic external to the e200z4204n3 core. This port provides information corresponding to each read or write access completed without error by the CPU. In order to report data accesses, the Nexus 3 DTC register DI control bit must be set to trace data accesses, not instruction accesses (i.e. the

normal setting. See the "Data Trace Control Register (DTC)" section in the Core (e200z4204n3) Nexus 3 Module chapter. The data trace port will report aligned accesses and misaligned accesses as one access, unless the two portions of a misaligned access are non-contiguous, such as when the second portion of the misaligned access is to address 0.

65.1.3 Hardware Debug Facilities

The e200z4204n3 core contains facilities that allow for external test and debugging. A modified IEEE 1149.1 control interface is used to communicate with the core resources. This interface is implemented through a standard 1149.1 TAP (test access port) controller.

By using public instructions, the external debugger can freeze or halt the e200z4204n3 core, read and write internal state and debug facilities, single-step instructions, and resume normal execution.

Hardware Debug is enabled by setting the External Debug Mode enable bit in External Debug Control register 0 ($EDBCR0_{EDM}$), which is also aliased to $DBCR0_{EDM}$. Setting $EDBCR0_{EDM}$ overrides the Internal Debug Mode enable bit $DBCR0_{IDM}$ unless resources are provided back to software via the settings in EDBRAC0. When the Hardware Debug facility is enabled, software is blocked from modifying the "hardware-owned" debug facilities. In addition, since resources are "owned" by the hardware debugger, inconsistent values may be present if software attempts to read "hardware-owned" debug-related resources.

When hardware debug is enabled by setting $EDBCR0_{EDM}$=1, the control registers and resources described in Debug registers are reserved for use by the external debugger. The same events described in Software Debug Events and Exceptions are also used for external debugging, but exceptions are not generated to running software. Hardware-owned debug events enabled in the respective DBCR0–8 registers are recorded in the EDBSR0 register (not the DBSR) regardless of MSR_{DE}, and no debug interrupts are generated unless a) the resource is granted back to software via EDBRAC0 settings, and b) debug mode entry is not masked by the corresponding event bit in EDBSRMSK0. Instead, the CPU will enter debug mode when an enabled event causes a EDBSR0 bit to become set. $EDBCR0_{EDM}$, EDBSR0, EDBSRMSK0, EDDEAR, and EDBRAC0 may only be written through the OnCE port.

Access to most debug resources (registers) requires that the CPU clock (**m_clk**) be running in order to perform write accesses from the external hardware debugger.

65.1.4 Sharing Debug Resources by Software/Hardware

Debug resources may be shared by a hardware debugger and software debug based on the settings of debug control register EDBRAC0. When $EDBCR0_{EDM}$ is set, EDBRAC0 settings determine which debug resources are allocated to software and which resources remain under exclusive hardware control. Software-owned resources which set DBSR bits when $DBCR0_{IDM}$=1 will cause a debug interrupt to occur when enabled with MSR_{DE}. Hardware-owned resources which set EDBSR0 bits when $EDBCR0_{EDM}$=1 will cause an entry into debug mode if the event is not masked in EDBSRMSK0. EDBRAC0 is read-only by software. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Hardware-owned resources will set status bits in the EDBSR0 register instead of in DBSR, and will report the effective address of data breakpoints in EDDEAR instead of DDEAR. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers are performed.

65.1.4.1 Simultaneous Hardware and Software Debug Event Handing

Since it is possible that a "hardware-owned" resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented which guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

The following figure shows the e200z4204n3 debug resources.

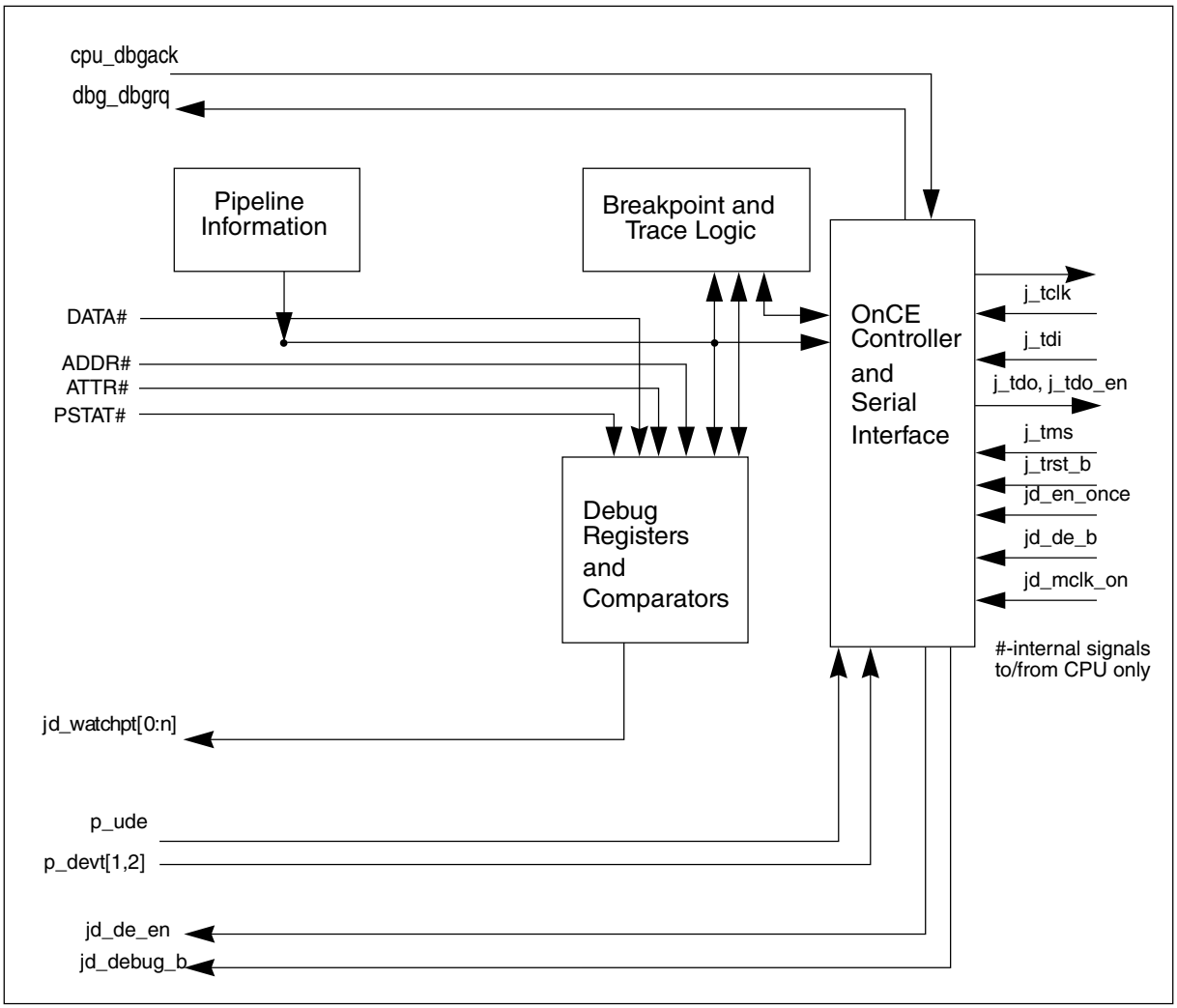


Figure 65-1. e200z4204n3 Debug Resources

65.2 Software Debug Events and Exceptions

Software debug events and exceptions are available when internal debug mode is enabled ($DBCR0_{IDM}$=1) and not overridden by external debug mode ($EDBCR0_{EDM}$ must either be cleared or corresponding resources must be allocated to software debug by the settings in EDBRAC0). When enabled, debug events cause debug exceptions to be recorded in the Debug Status Register. Specific event types are enabled by the Debug Control Registers (DBCR0–8). The Unconditional Debug Event (UDE) is an exception to this rule; it is always enabled. Once a Debug Status Register (DBSR) bit is set by a debug resource which is "owned" by software (other than MRR, DAC_OFST, or VLES), if Debug interrupts are enabled by MSR_{DE}, a Debug interrupt will be generated. The debug interrupt handler is responsible for ensuring that multiple repeated debug interrupts do not occur by clearing the DBSR as appropriate.

Certain debug events are not allowed to occur when MSR_{DE}=0 and $DBCR0_{IDM}$=1. In such situations, no debug exception occurs and thus no DBSR bit is set. Other debug events may cause debug exceptions and set DBSR bits regardless of the state of MSR_{DE}. A Debug interrupt will be delayed until MSR_{DE} is later set to '1'.

When a Debug Status Register bit is set while MSR_{DE}=0, an Imprecise Debug Event flag ($DBSR_{IDE}$) will also be set to indicate that an exception bit in the Debug Status Register was set while Debug interrupts were disabled. Debug interrupt handler software can use this bit to determine whether the address recorded in Debug Save/Restore Register 0 is an address associated with the instruction causing the debug exception, or the address of the instruction which enabled a delayed Debug interrupt by setting the MSR_{DE} bit. A **mtmsr** or **mtdbcr0** which causes both MSR_{DE} and $DBCR0_{IDM}$ to become set, enabling precise debug mode, may cause an Imprecise (Delayed) Debug exception to be generated due to an earlier recorded event in the Debug Status register.

There are eight types of debug events defined by *PowerISA 2.06*:

1. Instruction Address Compare debug events
2. Data Address Compare debug events
3. Trap debug events
4. Branch Taken debug events
5. Instruction Complete debug events
6. Interrupt/Critical Interrupt Taken debug events
7. Return/Critical Return debug events
8. Unconditional debug events

In addition, e200z4204n3 defines additional debug events:

- The External debug events DEVT1 and DEVT2 which are described in External debug event.
- The Performance Monitor Interrupt event PMI which is described in Performance Monitor Interrupt debug event.

The e200z4204n3 debug configuration supports most of these event types. Unsupported *PowerISA 2.06* defined functionality is as follows:

- Instruction Address Compare and Data Address Compare *Real address* mode is not supported

A brief description of each of the event types follows.

65.2.1 Instruction Address Compare Event

Instruction Address Compare debug events occur when enabled and execution is attempted of an instruction at an address that meets the criteria specified in the DBCR0, DBCR1, DBCR5, DBCR6, and IAC1–8 Registers. Instruction Address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. IAC events will not occur when an instruction would not have normally begun execution due to a higher priority exception at an instruction boundary.

IAC compares perform a 31-bit compare for VLE instruction storage accesses. Each halfword fetched by the instruction fetch unit will be marked with a set of bits indicating whether an Instruction Address Compare occurred on that halfword. Debug exceptions will occur if enabled and a 16-bit instruction, or the first halfword of a 32-bit instruction, is tagged with an IAC hit.

65.2.2 Data Address Compare Event

Data Address Compare debug events occur when enabled and execution of a load or store class instruction or a cache maintenance instruction results in a data access that meets the criteria specified in the DBCR0, DBCR2, DBCR4, DBCR7–8, DAC1–4, DVC1, and DVC2 Registers. Data address compares may specify user/supervisor mode, along with an effective address, masked effective address, or range of effective addresses for comparison. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. Four address compare values (DAC1–4) are provided.

The effective address of the load, store, or cache control operation is recorded in the DDEAR register when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set.

Note

In contrast to the *PowerISA 2.06* definition, Data Address Compare events on e200z4204n3 do not prevent the load or store class instruction from completing. If a load or store class

instruction completes successfully without a Data Storage interrupt, Data Address Compare exceptions are reported at the completion of the instruction. If the exception results in a precise Debug interrupt, the address value saved in DSRR0 is the address of the instruction following the load or store class instruction. For DVC DAC events, the exception can be imprecisely reported even further past the load or store class instruction generating the event (without necessarily affecting $DBSR_{IDE}$) and the saved address value can point to a subsequent instruction past the next instruction. This occurrence is indicated in the $DBSR_{DAC_OFST}$ field.

If a load or store class instruction does not complete successfully due to a Data Storage exception or a machine check condition for the load or store, and a Data Address Compare debug exception also occurs, the result is an imprecise Debug interrupt, the address value saved in DSRR0 is the address of the load or store class instruction, and the $DBSR_{IDE}$ bit will be set. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note

DAC events will not be recorded if a load multiple word or store multiple word instruction is interrupted prior to completion by a critical input or external input interrupt.

Note

DAC events will occur for cache control instructions without performing data compares, regardless of the settings of the DBCRx registers and DVC registers for data value comparison qualifications, i.e. no data comparisons are performed since these instructions do not have associated data values.

Note

DAC events are not signaled on the second portion of a misaligned load or store that is broken up into two separate accesses.

Note

DAC[1,2] events are not signaled if DVC[1,2]M is non-zero and a DSI exception occurs on the load or store, since the load or store access is not performed. For a **lmw** or **stmw** transfer however, if a DVC successfully occurs on a transfer and a later transfer encounters a DSI exception, the DAC event will be reported, since a successful data value compare took place.

Note

Due to internal pipeline status, for a reported DAC event on a **lmw** or **stmw** transfer, the value stored in the DDEAR/EDDEAR will be the effective address of the first transfer of the sequence, and not necessarily the precise transfer that caused the DAC event.

65.2.2.1 Data Address Compare Event Status Updates

Data Address Compare debug events with Data Value compares can be reported ambiguously in several circumstances involving issuing a sequence of load or store class instructions. Due to the CPU pipeline and the delay in performing the data value compare following completion of the access, if the first load or store class instruction generates a DVC DAC, a second and possibly third load or store class instruction may also generate a DAC or DVC DAC event, or may generate a DSI exception with or without a simultaneous DAC event.

Also, since non-load/store instructions may be dual-issued in combination with a load/store instruction, the actual number of additional instructions that are completed following a recognized DVC DAC on a load/store instruction may vary from 0 to 5. This value will be reported in the $DBSR_{DAC_OFST}$ field when the DVC DAC status is recorded.

Table 65-1 outlines the settings of the DBSR, DSRR0 saved value, and potential updating of the ESR register for various exception cases on sequences of load/store class instructions. Not all exception combinations are covered in the table, such as IAC, ISI, or Alignment exceptions on subsequent instructions. In general these exceptions will cause further instruction issue to be halted, execution of the excepting instruction to be aborted, and reporting of these exceptions will be masked. The saved DSRR0 value will point to this excepting instruction, and the exception(s) may be regenerated after returning from the debug interrupt handler and attempting to re-execute the instruction pointed to by

DSRR0. In addition, in the examples in Table 65-1, the DAC_OFST and DSRR0 values assume no dual issue occurs. If dual-issue occurs with the first, second, or third column, then the DAC_OFST and DSRR0 values will point beyond the values shown.

Table 65-1. DAC events and Resultant Updates

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
DSI, no DAC	—	—	Take DSI exception, no DBSR update. Update ESR.
DSI, with DACx	—	—	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST not set. DSRR0 points to 1st load/store class instruction. No ESR update.
DACx	—	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update.
DVC DACx	No exceptions, any instruction	No exceptions, Non-ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. No ESR update.
DVC DACx	No exceptions	No exceptions, Ldst instruction	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b010. DSRR0 points to instruction after 3rd instruction. No ESR update.
DVC DACx	DSI, no DAC	—	Take Debug exception, DBSR update setting DACx, DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DSI, with DACy	—	Take Debug exception, DBSR update setting DACx. DAC_OFST not set. DSRR0 points to 2nd load/store class instruction. No ESR update. **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DACy	—	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Non-Ldst instruction	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	Ldst instruction,no exception	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Normal Ldst	DSI Error, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. No ESR update. DSRR0 points to 3rd instruction.

Table continues on the next page...

Table 65-1. DAC events and Resultant Updates (continued)

1st load/store class instruction	2nd instruction (load/store class unless otherwise specified)	3rd instruction (load/store class unless otherwise specified)	Result
			Note: in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	DVC DACy,Normal Ldst	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd load/store class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	DVC DACy,Ldst multiple (lmw, stmw)	Any instruction including ld/st	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
DVC DACx	Any instruction(no exception)	DSI, with or without DAC,Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx. DAC_OFST set to 3'b001. DSRR0 points to the 3rd instruction. No ESR update. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
DVC DACx	Any instruction(no exception)	DACy, or DVC DAC_y Normal Ldst or multiple word Ldst	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction after the 3rd class instruction. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 65-2 through Table 65-5 show some example updates for specific code sequences of dual issuing of load/store class instructions with non-load/store class instructions and the results of DAC and DVC events on selected ones of the load/store instructions.

Table 65-2. DAC events and Resultant Updates, Dual-issue case 1

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1):	Take DSI exception, no DBSR update, Update ESR.

Table continues on the next page...

Table 65-2. DAC events and Resultant Updates, Dual-issue case 1 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	DSI, no DAC	
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. No ESR update. DSRR0 points to instruction (6).

Table 65-2. DAC events and Resultant Updates, Dual-issue case 1

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (3): DVC DACy Instruction (5): DACy or DVC DACy	**Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 65-3. DAC events and Resultant Updates, Dual-issue case 2

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction after instruction (6). No ESR update.
	Instruction (1): DVC DACx Instruction (3):	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.

Table continues on the next page...

Table 65-3. DAC events and Resultant Updates, Dual-issue case 2 (continued)

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) load/store	Event(s)	Result
	DSI, with or without DAC	
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b101. DSRR0 points to instruction (7). No ESR update. **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACyInstruction (6): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. No ESR update. DSRR0 points to instruction (4). No debug event updates for instruction (4). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case. **Note:** in this case the 3rd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DVC DACy Instruction (6): DACy or DVC DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b101. No ESR update. DSRR0 points to instruction (7). No debug event updates for instruction (7). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 65-4. DAC events and Resultant Updates, Dual-issue case 3

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) alu (4) alu (5) load/store (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6).
	Instruction (1): DVC DACx Instruction (5): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (5): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No debug event updates for instruction (6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (5): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b100. DSRR0 points to instruction (6). No ESR update. No debug event updates for instruction (6) **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

Table 65-5. DAC events and Resultant Updates, Dual-issue case 4

Instruction Sequence: The following pairs dual-issue: (1) load/store (2) alu (3) load/store (4) alu (5) alu (6) alu	Event(s)	Result
	Instruction (1): DSI, no DAC	Take DSI exception, no DBSR update, Update ESR.
	Instruction (1): DSI, with DACx	Take Debug exception, DBSR update setting DACx and IDE, DAC_OFST set to 3'b000. DSRR0 points to instruction (1). No ESR update.
	Instruction (1): DACx	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b000. DSRR0 points to instruction (2). No ESR update.
	Instruction (1): DVC DACx No other exceptions	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6)
	Instruction (1): DVC DACx Instruction (3): DSI, with or without DAC	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b001. DSRR0 points to instruction (3). No ESR update. No debug event updates for instructions (3)–(6). **Note:** in this case the 2nd ld/st exception is masked. This behavior is implementation dependent and may differ on other CPUs.
	Instruction (1): DVC DACx Instruction (3): DACy	Take Debug exception, DBSR update setting DACx, DACy. DAC_OFST set to 3'b010. DSRR0 points to instruction (4). No debug event updates for instructions (4)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.
	Instruction (1): DVC DACx Instruction (3): DVC DACy	Take Debug exception, DBSR update setting DACx, DAC_OFST set to 3'b011. DSRR0 points to instruction (5). No ESR update. No debug event updates for instructions (5)–(6). **Note:** in this case if x==y, then the resultant state of DBSR and DSRR0 may be indistinguishable from the "no DACy" case.

65.2.3 Linked Instruction Address Compare and Data Address Compare Events

Data Address Compare 1, 2, 3, and 4 debug events may be 'linked' with an Instruction Address Compare event by setting the DAC[1–4]LNK control bits in DBCR2 and DBCR8 to further refine when a Data Address Compare debug event is generated. DAC1

may be linked with IAC1, DAC2 (when not used as a mask or range bounds register) may be linked with IAC3. DAC3 may be linked with IAC5, DAC4 (when not used as a mask or range bounds register) may be linked with IAC7. When linked, a DAC debug event occurs when the same instruction that generates the DAC 'hit' also generates a corresponding linked IAC 'hit'. When linked, the IAC event is not recorded in the Debug Status register, regardless of whether a corresponding linked DAC event occurs, or whether the IAC event enable is set.

When enabled and execution of a load or store class instruction results in a data access with an address that meets the criteria specified in the DBCRx, DACx, and DVCx Registers, and the instruction also meets the criteria for generating an Instruction Address Compare event, a Linked Data Address Compare debug event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. The normal DAC1 status bit in the DBSR is used for recording these events. The IAC status bit is not set if the corresponding Instruction Address Compare register is linked.

Linking is enabled using control bits in DBCR2 and DBCR8. Note that linking is only available in EDM or IDM. Attempts to use linking otherwise are ignored.

Note

Linked DAC events will not be recorded if a load multiple word or store multiple word type instruction is interrupted prior to completion by a critical input or external input interrupt.

65.2.4 Trap Debug Event

A Trap debug event (TRAP) occurs if Trap debug events are enabled ($DBCR0_{TRAP}$=1), a Trap instruction (**tw**) is executed, and the conditions specified by the instruction for the trap are met. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Trap debug event occurs, the $DBSR_{TRAP}$ bit is set to 1 to record the debug exception.

65.2.5 Branch Taken Debug Event

A Branch Taken debug event (BRT) occurs if Branch Taken debug events are enabled ($DBCR0_{BRT}$=1) and execution is attempted of a branch instruction that will be taken (either an unconditional branch, or a conditional branch whose branch condition is true), and MSR_{DE}=1. Branch Taken debug events are not recognized if MSR_{DE}=0 at the time of execution of the branch instruction and thus $DBSR_{IDE}$ can not be set by a Branch Taken

debug event. When a Branch Taken debug event is recognized, the $DBSR_{BRT}$ bit is set to 1 to record the debug exception, and the address of the branch instruction will be recorded in DSRR0.

65.2.6 Instruction Complete Debug Event

An Instruction Complete debug event (ICMP) occurs if Instruction Complete debug events are enabled ($DBCR0_{ICMP}$=1), execution of any instruction is completed, and MSR_{DE}=1. If execution of an instruction is suppressed due to the instruction causing some other exception that is enabled to generate an interrupt, then the attempted execution of that instruction does not cause an Instruction Complete debug event. The ***se_sc*** instruction does not fall into the category of an instruction whose execution is suppressed, since the instruction actually executes and then generates a System Call interrupt. In this case, the Instruction Complete debug exception will also be set. When an Instruction Complete debug event is recognized, $DBSR_{ICMP}$ is set to 1 to record the debug exception and the address of the next instruction to be executed will be recorded in DSRR0.

Instruction Complete debug events are not recognized if MSR_{DE}=0 at the time of execution of the instruction, thus $DBSR_{IDE}$ is not generally set by an ICMP debug event.

One circumstance may cause the $DBSR_{ICMP}$ and $DBSR_{IDE}$ bits to be set. This occurs when a EFPU Round exception occurs. Since the instruction is by definition completed (SRR0 points to the following instruction), this interrupt takes higher priority than the Debug interrupt so as not to be lost, and $DBSR_{IDE}$ is set to indicate the imprecise recognition of a Debug interrupt. In this case, the Debug interrupt will be taken with SRR0 pointing to the instruction following the instruction that generated the EFPU Round exception, and DSRR0 will point to the Round exception handler. In addition to occurring when $DBCR0_{IDM}$=1, this circumstance can also occur when $EDBCR0_{EDM}$=1 and the event is hardware-owned, in which case $EDBSR0_{IDE}$ will be set.

Note

Instruction complete debug events are not generated by the execution of an instruction that sets MSR_{DE} to '1' while $DBCR0_{ICMP}$=1, nor by the execution of an instruction that sets $DBCR0_{ICMP}$ to '1' while MSR_{DE}=1.

65.2.7 Interrupt Taken Debug Event

An Interrupt Taken debug event (IRPT) occurs if Interrupt Taken debug events are enabled ($DBCR0_{IRPT}$=1) and a base-class interrupt occurs. Only base-class interrupts (an interrupt using SRR0/1) cause an Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Interrupt Taken debug event occurs, the $DBSR_{IRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the non-critical interrupt handler.

65.2.8 Critical Interrupt Taken Debug Event

A Critical Interrupt Taken debug event (CIRPT) occurs if Critical Interrupt Taken debug events are enabled ($DBCR0_{CIRPT}$=1) and a critical interrupt occurs. Only critical class interrupts (an interrupt using CSRR0/1) cause a Critical Interrupt Taken debug event. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Interrupt Taken debug event occurs, the $DBSR_{CIRPT}$ bit is set to 1 to record the debug exception. The value saved in DSRR0 will be the address of the critical interrupt handler.

65.2.9 Return Debug Event

A Return debug event (RET) occurs if Return debug events are enabled ($DBCR0_{RET}$=1) and an attempt is made to execute an **se_rfi** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Return debug event occurs, the $DBSR_{RET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfi** (i.e. before the MSR is updated by the **se_rfi**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfi**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfi** instruction.

65.2.10 Critical Return Debug Event

A Critical Return debug event (CRET) occurs if Critical Return debug events are enabled ($DBCR0_{CRET}$=1) and an attempt is made to execute an **se_rfci** instruction. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Critical Return debug event occurs, the $DBSR_{CRET}$ bit is set to 1 to record the debug exception.

If MSR_{DE}=0 at the time of the execution of the **se_rfci** (i.e. before the MSR is updated by the **se_rfci**), then $DBSR_{IDE}$ is also set to 1 to record the imprecise debug event.

If MSR_{DE}=1 at the time of the execution of the **se_rfci**, a Debug interrupt will occur provided there exists no higher priority exception that is enabled to cause an interrupt. Debug Save/Restore Register 0 will be set to the address of the **se_rfci** instruction.

65.2.11 External debug event

An External debug event (DEVT1, DEVT2) occurs if External debug events are enabled ($DBCR0_{DEVT1}$=1 or $DBCR0_{DEVT2}$=1), and the respective **p_devt1** or **p_devt2** input signal transitions to the asserted state while the CPU is not in the Stopped state. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an External debug event occurs, $DBSR_{DEVT\{1,2\}}$ is set to '1' to record the debug exception. This debug event is an asynchronous event, but is only sampled when the CPU **m_clk** is active.

65.2.12 Unconditional debug event

An Unconditional debug event (UDE) occurs when the Unconditional Debug Event (**p_ude**) input transitions to the asserted state. The Unconditional debug event is the only debug event that does not have a corresponding enable bit for the event in DBCR0. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When an Unconditional debug event occurs, the $DBSR_{UDE}$ bit is set to '1' to record the debug exception. This debug event is an asynchronous event.

65.2.13 Performance Monitor Interrupt debug event

A Performance Monitor Interrupt debug event (PMI) occurs if Performance Monitor Interrupt debug events are enabled ($PMGC0_{UDI}$=1), and a performance monitor interrupt event occurs. This event can occur and be recorded in DBSR regardless of the setting of MSR_{DE}. When a Performance Monitor Interrupt debug event occurs, $DBSR_{PMI}$ is set to '1' to record the debug exception. This debug event is an asynchronous event.

65.3 Debug registers

This section describes debug-related registers that are software accessible. These registers are intended for use by special debug tools and debug software, not by general application code.

Access to these registers (other than DBSR) by software is conditioned by the External Debug mode control bit ($EDBCR0_{EDM}$) and the settings of debug control register EDBRAC0, which can be set by the hardware debug port. If $EDBCR0_{EDM}$ is set and if the bit in EDBRAC0 corresponding to the resource is cleared, software is prevented from modifying debug register values other than in DBSR, since the resource is not "owned" by software. Software always has ownership of DBSR. Execution of a **mtspr** instruction targeting a debug register or register field not "owned" by software will not cause modifications to occur, and no exception will be signaled. In addition, since the external debugger hardware may be manipulating debug register values, the state of these registers or register fields not "owned" by software is not guaranteed to be consistent if accessed (read) by software with a **mfspr** instruction, other than the $DBCR0_{EDM}$ bit itself and the EDBRAC0 register. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control registers as appropriate when hardware modifications to the debug registers is performed.

65.3.1 Debug Address and Value Registers

Instruction Address Compare registers IAC1–8 are used to hold instruction addresses for address comparison purposes. In addition, IAC2 and IAC4 may hold mask information for IAC1 and IAC3 respectively and IAC6 and IAC8 may hold mask information for IAC5 and IAC7 respectively, when *Address Bit Match* compare modes are selected. Note that when performing instruction address compares, the low order bit of the instruction address and the corresponding IAC register is ignored for VLE instructions.

Data Address Compare registers DAC1–4 are used to hold data access addresses for address comparison purposes. In addition, DAC2 and DAC4 may hold mask information for DAC1 and DAC3 respectively when *Address Bit Match* compare modes are selected.

Data Value Compare registers DVC1 and DVC2 are used to hold data values for data comparison purposes. DVC1 and DVC2 are 64-bit registers. Data value comparisons are used to qualify Data Address compare 1 and 2 debug events. Data value comparisons are

not available for Data address compare 3–4 events. DVC1 is associated with DAC1, and DVC2 is associated with DAC2. The most significant byte of the DVC1(2) register (labeled B0 in the following figure) corresponds to the byte data value transferred to/from memory byte offset 0, 8, …, and the least significant byte of the register (labeled B7 in the following figure) corresponds to byte offset 7, F, … . When enabled for performing data value comparisons, each enabled byte in DVC1(2) is compared with the memory value transferred on the corresponding active byte lane of the data memory interface to determine if a match occurs. Inactive byte lanes do not participate in the comparison, they are implicitly masked. The Byte Strobe Assertion for Transfers table in the Core (e200z4204n3) Core Complex Overview chapter shows active byte lanes for data transfers. Software must also program the DVC1(2) register byte positions based on the alignment of the access. Misaligned accesses are not fully supported, since the data address and data value comparisons are only performed on the initial access in the case of a misaligned access; thus, accesses that cross a 64-bit boundary cannot be fully matched. For address and size combinations that involve two transfers, only the initial transfer is used for data address and value matching.

DVC1 and DVC2 may be read or written using **mtspr** and **mfspr** instructions. The DVC1U and DVC2U SPR numbers (601,602) are used for accessing the upper 32-bit portion of the DVC register. The DVC1 and DVC2 SPR numbers (318,319) are used for accessing the lower 32-bit portion of the DVC register.

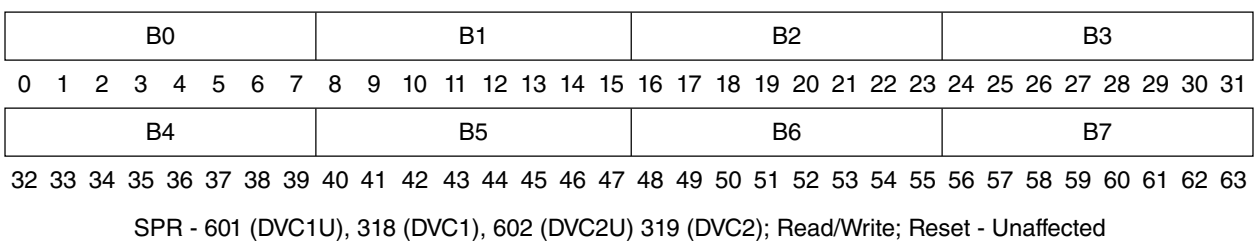


Figure 65-2. DVC1, DVC2 registers

65.3.2 Debug Control and Status registers

Debug Control Registers (DBCR0–8 and EDBRAC0) are used to enable debug events, reset the processor, and set the debug mode of the processor. The Debug Status register (DBSR) records debug exceptions while Internal Debug mode is enabled. The Debug Data Effective Address register (DDEAR) records the effective address of a Data Address Compare event while Internal Debug mode is enabled.

e200z4204n3 requires that a context synchronizing instruction follow a **mtspr** DBCR0–8 or DBSR to ensure that any alterations enabling/disabling debug events are effective. The context synchronizing instruction may or may not be affected by the alteration. Typically, an **se_isync** instruction is used to create a synchronization boundary beyond which it can be guaranteed that the newly written control values are in effect.

For watchpoint generation, configuration settings contained in DBCR1–8 are used, even though the corresponding event(s) may be disabled (via DBCR0) from setting DBSR flags.

65.3.2.1 Debug Control Register 0 (DBCR0)

Debug Control Register 0 is used to enable debug modes and controls which debug events are allowed to set DBSR or EDBSR0 flags. e200z4204n3 adds some implementation specific bits to this register, as seen in the following figure.

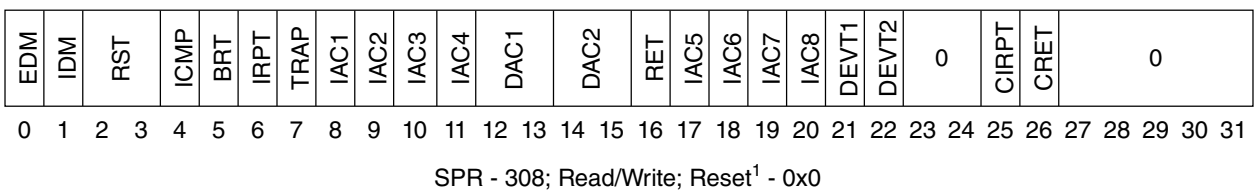


Figure 65-3. Debug Control Register 0 (DBCR0) register

Note

[1] $DBCR0_{EDM}$ is affected by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b**. All other bits are reset by processor reset **p_reset_b** if $DBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $DBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources (other than RST) from reset by **p_reset_b**, and only software-owned resources indicated by EDBRAC0 and the $DBCR0_{RST}$ field will be reset by **p_reset_b**. The $DBCR0_{RST}$ field will always be reset by **p_reset_b** regardless of the value of $DBCR0_{EDM}$.

The following provides bit definitions for Debug Control Register 0.

Table 65-6. DBCR0 field descriptions

Bit	Name	Description
0	EDM	External Debug mode. This bit is read-only by software. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset p_reset_b. This allows the debugger to set up hardware debug events that remain active across a processor reset. 0 External debug mode disabled. Internal debug events not mapped into external debug events. 1 External debug mode enabled. Hardware-owned debug events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCR0–8, IAC1–8, DAC1–4, DVC1–2[U]} unless permitted by settings in EDBRAC0. Hardware-owned events will set status bits in EDBSR0. Programming Notes: It is recommended that debug status bits in the Debug Status Registers be cleared before disabling external debug mode to avoid any internal imprecise debug interrupts. Software may use this bit to determine if external debug has control over the debug registers. The hardware debugger must set the EDM bit to '1' before other bits in this register (and other debug registers) may be altered. On the initial setting of this bit to '1', all other bits are unchanged. This bit is only writable through the OnCE port.
1	IDM	Internal Debug mode 0 Debug exceptions are disabled. Debug events do not affect DBSR. 1 Debug exceptions are enabled. Enabled debug events owned by software will update the DBSR. If MSR_{DE}=1, the occurrence of a debug event, or the recording of an earlier debug event in the Debug Status Register when MSR_{DE} was cleared, will cause a Debug interrupt.
2:3	RST	Reset Control 00 No function 01 **p_dbrstc[1]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset 10 **p_dbrstc[0]** pin asserted by Debug Reset Control. Allows external device to initiate processor or system reset. 11 Reserved
4	ICMP	Instruction Complete Debug Event Enable 0 ICMP debug events are disabled 1 ICMP debug events are enabled
5	BRT	Branch Taken Debug Event Enable 0 BRT debug events are disabled 1 BRT debug events are enabled
6	IRPT	Interrupt Taken Debug Event Enable 0 IRPT debug events are disabled 1 IRPT debug events are enabled
7	TRAP	Trap Taken Debug Event Enable 0 TRAP debug events are disabled 1 TRAP debug events are enabled
8	IAC1	Instruction Address Compare 1 Debug Event Enable 0 IAC1 debug events are disabled

Table continues on the next page...

Table 65-6. DBCR0 field descriptions (continued)

Bit	Name	Description
		1 IAC1 debug events are enabled
9	IAC2	Instruction Address Compare 2 Debug Event Enable 0 IAC2 debug events are disabled 1 IAC2 debug events are enabled
10	IAC3	Instruction Address Compare 3 Debug Event Enable 0 IAC3 debug events are disabled 1 IAC3 debug events are enabled
11	IAC4	Instruction Address Compare 4 Debug Event Enable 0 IAC4 debug events are disabled 1 IAC4 debug events are enabled
12:13	DAC1	Data Address Compare 1 Debug Event Enable 00 DAC1 debug events are disabled 01 DAC1 debug events are enabled only for store-type data storage accesses 10 DAC1 debug events are enabled only for load-type data storage accesses 11 DAC1 debug events are enabled for load-type or store-type data storage accesses
14:15	DAC2	Data Address Compare 2 Debug Event Enable 00 DAC2 debug events are disabled 01 DAC2 debug events are enabled only for store-type data storage accesses 10 DAC2 debug events are enabled only for load-type data storage accesses 11 DAC2 debug events are enabled for load-type or store-type data storage accesses
16	RET	Return Debug Event Enable 0 RET debug events are disabled 1 RET debug events are enabled
17	IAC5	Instruction Address Compare 5 Debug Event Enable 0 IAC5 debug events are disabled 1 IAC5 debug events are enabled
18	IAC6	Instruction Address Compare 6 Debug Event Enable 0 IAC6 debug events are disabled 1 IAC6 debug events are enabled
19	IAC7	Instruction Address Compare 7 Debug Event Enable 0 IAC7 debug events are disabled 1 IAC7 debug events are enabled
20	IAC8	Instruction Address Compare 8 Debug Event Enable 0 IAC8 debug events are disabled 1- IAC8 debug events are enabled
21	DEVT1	External Debug Event 1 Enable 0 DEVT1 debug events are disabled 1 DEVT1 debug events are enabled

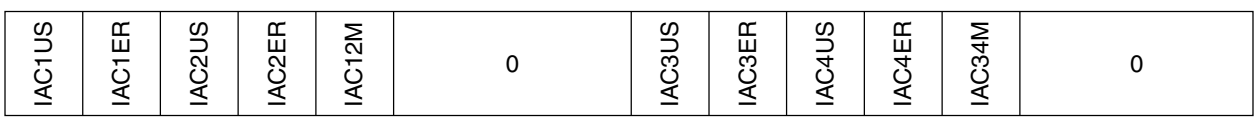
Table continues on the next page...

Table 65-6. DBCR0 field descriptions (continued)

Bit	Name	Description
22	DEVT2	External Debug Event 2 Enable 0 DEVT2 debug events are disabled 1 DEVT2 debug events are enabled
23:24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Enable 0 CIRPT debug events are disabled 1 CIRPT debug events are enabled
26	CRET	Critical Return Debug Event Enable 0 CRET debug events are disabled 1 CRET debug events are enabled
27:31	—	Reserved

65.3.2.2 Debug Control Register 1 (DBCR1)

Debug Control Register 1 is used to configure Instruction Address Compare operation. The DBCR1 register is shown in the following figure.



SPR - 309; Read/Write; Reset[1] - 0x0

Figure 65-4. DBCR1 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 1.

Table 65-7. DBCR1 field descriptions

Bit	Name	Description
0:1	IAC1US	Instruction Address Compare 1 User/Supervisor Mode

Table continues on the next page...

Table 65-7. DBCR1 field descriptions (continued)

Bit	Name	Description
		00 IAC1 debug events not affected by MSR_{PR}
		01 Reserved
		10 IAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode)
		11 IAC1 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC1ER	Instruction Address Compare 1 Effective/Real Mode
		00 IAC1 debug events are based on effective address
		01 Unimplemented (Book E real address compare), no match can occur
		10 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=0
		11 IAC1 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC2US	Instruction Address Compare 2 User/Supervisor Mode
		00 IAC2 debug events not affected by MSR_{PR}
		01 Reserved
		10 IAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode)
		11 IAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC2ER	Instruction Address Compare 2 Effective/Real Mode
		00 IAC2 debug events are based on effective address
		01 Unimplemented (Book E real address compare), no match can occur
		10 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=0
		11 IAC2 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC12M	Instruction Address Compare 1/2 Mode
		00 Exact address compare. IAC1 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC1. IAC2 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC2.
		01 Address bit match. IAC1 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC2 are equal to the contents of IAC1, also ANDed with the contents of IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
		10 Inclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC1 and less than the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
		11 Exclusive address range compare. IAC1 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC1 or is greater than or equal to the value specified in IAC2. IAC2 debug events do not occur. IAC1US and IAC1ER settings are used.
10:15	—	Reserved
16:17	IAC3US	Instruction Address Compare 3 User/Supervisor Mode
		00 IAC3 debug events not affected by MSR_{PR}
		01 Reserved
		10 IAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode)
		11 IAC3 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC3ER	Instruction Address Compare 3 Effective/Real Mode
		00 IAC3 debug events are based on effective address

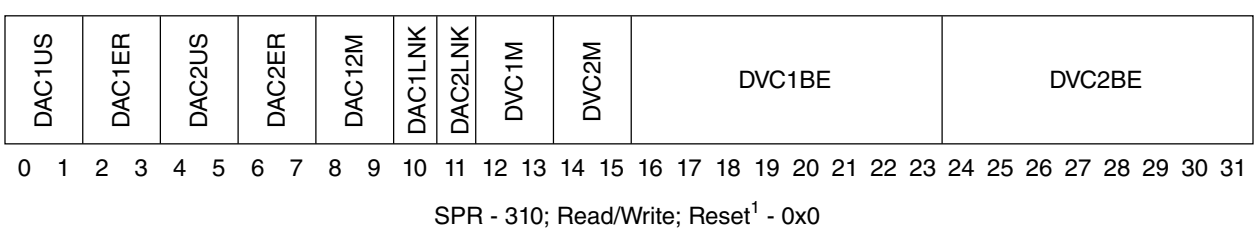
Table continues on the next page...

Table 65-7. DBCR1 field descriptions (continued)

Bit	Name	Description
		01 Unimplemented (Book E real address compare), no match can occur 10 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC3 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC4US	Instruction Address Compare 4 User/Supervisor Mode 00 IAC4 debug events not affected by MSR_{PR} 01 Reserved 10 IAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC4 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC4ER	Instruction Address Compare 4 Effective/Real Mode 00 IAC4 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC4 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC34M	Instruction Address Compare 3/4 Mode 00 Exact address compare. IAC3 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC3. IAC4 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC4. 01 Address bit match. IAC3 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC4 are equal to the contents of IAC3, also ANDed with the contents of IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 10 Inclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC3 and less than the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used. 11 Exclusive address range compare. IAC3 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC3 or is greater than or equal to the value specified in IAC4. IAC4 debug events do not occur. IAC3US and IAC3ER settings are used.
26:31	—	Reserved

65.3.2.3 Debug Control Register 2 (DBCR2)

Debug Control Register 2 is used to configure Data Address Compare and Data Value Compare operation.The DBCR2 register is shown in the following figure.



SPR - 310; Read/Write; Reset[1] - 0x0

Figure 65-5. DBCR2 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 2.

Table 65-8. DBCR2 Bit Definitions

Bit	Name	Description
0:1	DAC1US	Data Address Compare 1 User/Supervisor Mode 00 DAC1 debug events not affected by MSR_{PR} 01 Reserved 10 DAC1 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC1 debug events can only occur if MSR_{PR}=1 (User mode)
2:3	DAC1ER	Data Address Compare 1 Effective/Real Mode 00 DAC1 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC1 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC2US	Data Address Compare 2 User/Supervisor Mode 00 DAC2 debug events not affected by MSR_{PR} 01 Reserved 10 DAC2 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 DAC2 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC2ER	Data Address Compare 2 Effective/Real Mode 00 DAC2 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=0 11 DAC2 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC12M	Data Address Compare 1/2 Mode 00 Exact address compare. DAC1 debug events can only occur if the address of the data access is equal to the value specified in DAC1. DAC2 debug events can only occur if the address of the data access is equal to the value specified in DAC2. 01 Address bit match. DAC1 debug events can occur only if the address of the data access ANDed with the contents of DAC2, are equal to the contents of DAC1 also ANDed with the contents of DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used. 10 Inclusive address range compare. DAC1 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC1 and less than the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.

Table continues on the next page...

Table 65-8. DBCR2 Bit Definitions (continued)

Bit	Name	Description
		11 Exclusive address range compare. DAC1 debug events can occur only if the address of the data access is less than the value specified in DAC1 or is greater than or equal to the value specified in DAC2. DAC2 debug events do not occur. DAC1US and DAC1ER settings are used.
10	DAC1LNK	Data Address Compare 1 Linked. When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC1 debug events are linked to IAC1 debug events. IAC1 debug events do not affect DBSR When linked to IAC1, DAC1 debug events are conditioned based on whether the instruction also generated an IAC1 debug event. Note that linking is only available in EDM or IDM.
11	DAC2LNK	Data Address Compare 2 Linked. When linked to IAC3, DAC2 debug events are conditioned based on whether the instruction also generated an IAC3 debug event. DAC2 can only be linked if DAC12M specifies Exact Address Compare since DAC2 debug events are not generated in the other compare modes. Note that linking is only available in EDM or IDM. 0 No effect 1 DAC 2 debug events are linked to IAC3 debug events. IAC3 debug events do not affect DBSR
12:13	DVC1M	Data Value Compare 1 Mode When $DBCR4_{DVC1C}$=0: 00 DAC1 debug events not affected by data value compares. 01 DAC1 debug events can only occur when all bytes specified in the DVC1BE field match the corresponding data byte values for active byte lanes of the memory access. 10 DAC1 debug events can only occur when any byte specified in the DVC1BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC1 debug events can only occur when all bytes specified in the DVC1BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC1 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC1C}$=1: 00 Reserved 01 DAC1 debug events can only occur when any byte specified in the DVC1BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC1 debug events can only occur when all bytes specified in the DVC1BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Note: Inactive byte lanes of the memory access are automatically masked.
14:15	DVC2M	Data Value Compare 2 Mode When $DBCR4_{DVC2C}$=0: 00 DAC2 debug events not affected by data value compares. 01 DAC2 debug events can only occur when all bytes specified in the DVC2BE field match the corresponding data byte values for active byte lanes of the memory access.

Table continues on the next page...

Table 65-8. DBCR2 Bit Definitions (continued)

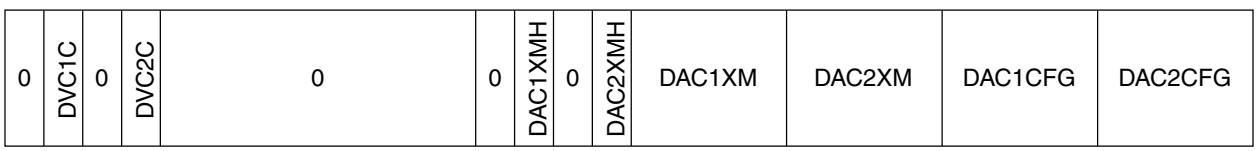
Bit	Name	Description
		10 DAC2 debug events can only occur when any byte specified in the DVC2BE field matches the corresponding data byte value for active byte lanes of the memory access. 11 DAC2 debug events can only occur when all bytes specified in the DVC2BE field within at least one of the halfwords of the data value of the memory access matches the corresponding DVC2 value. **NOTE:** Inactive byte lanes of the memory access are automatically masked. When $DBCR4_{DVC2C}$=1: 00 Reserved 01 DAC2 debug events can only occur when any byte specified in the DVC2BE field does not match the corresponding data byte value for active byte lanes of the memory access. If all active bytes match, then no event will be generated. 10 DAC2 debug events can only occur when all bytes specified in the DVC2BE field do not match the corresponding data byte values for active byte lanes of the memory access. If any active byte match occurs, no event will be generated. 11 Reserved **NOTE:** Inactive byte lanes of the memory access are automatically masked.
16:23	DVC1BE	Data Value Compare 1 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC1. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC1. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC1. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC1. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC1. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC1. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC1. xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC1. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC1.
24:31	DVC2BE	Data Value Compare2 Byte Enables Specifies which bytes in the aligned doubleword value associated with the memory access are compared to the corresponding bytes in DVC2. Inactive byte lanes of a memory access smaller than 64 bits are automatically masked by hardware. If all bits in the DVC1BE field are clear, then a match will occur regardless of the data. Misaligned accesses that cross a doubleword boundary are not fully supported. 1xxxxxxx Byte lane 0 is enabled for comparison with the value in bits 0:7 of DVC2. x1xxxxxx Byte lane 1 is enabled for comparison with the value in bits 8:15 of DVC2. xx1xxxxx Byte lane 2 is enabled for comparison with the value in bits 16:23 of DVC2. xxx1xxxx Byte lane 3 is enabled for comparison with the value in bits 24:31 of DVC2. xxxx1xxx Byte lane 4 is enabled for comparison with the value in bits 32:39 of DVC2. xxxxx1xx Byte lane 5 is enabled for comparison with the value in bits 40:47 of DVC2.

Table 65-8. DBCR2 Bit Definitions

Bit	Name	Description
		xxxxxx1x Byte lane 6 is enabled for comparison with the value in bits 48:55 of DVC2. xxxxxxx1 Byte lane 7 is enabled for comparison with the value in bits 56:63 of DVC2.

65.3.2.4 Debug Control Register 4 (DBCR4)

Debug Control Register 4 is used to extend data address and value compare matching functionality. DBCR4 is shown in the following figure.



SPR - 563; Read/Write; Reset[1] - 0x0

Figure 65-6. DBCR4 Register

Note

[1] DBCR4 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 4.

Table 65-9. DBCR4 field description

Bit	Name	Description
0	—	Reserved
1	DVC1C	Data Value Compare 1 Control. DVC1C controls whether DVC1 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition. 0 Normal DVC1 operation. Inverted polarity DVC1 operation
2	—	Reserved
3	DVC2C	Data Value Compare 2 Control. DVC2C controls whether DVC2 data value comparisons utilize the normal compare operation, or an alternate “inverted compare” operation. In inverted polarity mode, data value compares perform a not-equal comparison. See details in the DBCR2 register definition 0 Normal DVC2 operation.

Table continues on the next page...

Table 65-9. DBCR4 field description (continued)

Bit	Name	Description
		1 Inverted polarity DVC2 operation
4:12	—	Reserved
13	DAC1XMH	Data Address Compare 1 Extended Mask Control High. DAC1XMH extends the range of the DAC1XM field. 0 - DAC1XM masks 0–15 low-order address bits 1 - DAC1XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC2XMH	Data Address Compare 2 Extended Mask Control High. . DAC2XMH extends the range of the DAC2XM field. 0 DAC2XM masks 0–15 low-order address bits 1 DAC2XM masks 16–31 low-order address bits
16:19	DAC1XM	Data Address Compare 1 Extended Mask Control. DAC1XM allows for binary power of 2 address range compares for DAC1 without requiring the use of DAC2. Value of DAC1XMH || DAC1XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC1 when comparing the storage address with the value in DAC1 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC2XM	Data Address Compare 2 Extended Mask Control Value of DAC2XMH || DAC2XM: 00000 No additional masking when DBCR2[DAC12M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC2 when comparing the storage address with the value in DAC2 for exact address compare (DBCR2[DAC12M] = 00). Address ranges of 2 bytes to 2GB are supported. DAC2XM allows for binary power of 2 address range compares for DAC2.
24:27	DAC1CFG	Data Address Compare 1 Configuration 0000 DAC1 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC1}$=00 0001 DAC1 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC1}$=00 0010 DAC1 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC1}$=00 0011 Reserved 01xx Reserved 1000 DAC1 address comparisons are used for stack limit checking. DAC1 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC1. When a qualified DAC1 match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC1 watchpoint is generated. $DBCR0_{DAC1}$ and $DBCR2_{DVC1M}$ settings are ignored. 1001 – 1111 Reserved DAC1CFG controls whether DAC1 data address comparisons utilize the normal PowerISA operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint).

Table continues on the next page...

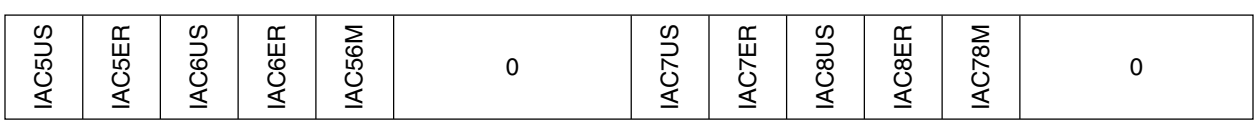
Table 65-9. DBCR4 field description (continued)

Bit	Name	Description
		NOTE: unlike the DAC3–4CFG fields, debug event enabling for DAC1 is controlled by the DAC1 field in DBCR0. The DAC1CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC1}$=0; when $DBCR0_{DAC1}$!=00, DAC1 watchpoints will fire whenever a DAC1 debug event occurs.
28:31	DAC2CFG	Data Address Compare 2 Configuration 0000 DAC2 debug watchpoints are enabled for load-type or store-type data storage accesses when $DBCR0_{DAC2}$=00 0001 DAC2 debug watchpoints are enabled only for store-type data storage accesses when $DBCR0_{DAC2}$=00 0010 DAC2 debug watchpoints are enabled only for load-type data storage accesses when $DBCR0_{DAC2}$=00 0011 Reserved 01xx Reserved 1xxx Reserved DAC2CFG controls whether DAC2 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. **NOTE:** Unlike the DAC3–4CFG fields, debug event enabling for DAC2 is controlled by the DAC2 field in DBCR0. The DAC2CFG encodings 0000–0010 are used to control watchpoint generation when $DBCR0_{DAC2}$=00. When $DBCR0_{DAC2}$!=00, DAC2 watchpoints will fire whenever a DAC2 debug event occurs.

65.3.2.5 Debug Control Register 5 (DBCR5)

Debug Control Register 5 is used to configure Instruction Address Compare operation for IAC5–8. The DBCR5 register is shown in the following figure.

.



SPR - 564; Read/Write; Reset[1] - 0x0

Figure 65-7. DBCR5 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 5.

Table 65-10. DBCR5 field descriptions

Bit(s)	Name	Description
0:1	IAC5US	Instruction Address Compare 5 User/Supervisor Mode 00 IAC5 debug events not affected by MSR_{PR} 01 Reserved 10 IAC5 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC5 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	IAC5ER	Instruction Address Compare 5 Effective/Real Mode 00 IAC5 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC5 debug events are based on effective address and can only occur if MSR_{IS}=1
4:5	IAC6US	Instruction Address Compare 6 User/Supervisor Mode 00 IAC6 debug events not affected by MSR_{PR} 01 Reserved 10 IAC6 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC6 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	IAC6ER	Instruction Address Compare 6 Effective/Real Mode 00 IAC6 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC6 debug events are based on effective address and can only occur if MSR_{IS}=1
8:9	IAC56M	Instruction Address Compare 5/6 Mode 00 Exact address compare. IAC5 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC5. IAC6 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC6. 01 Address bit match. IAC5 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC6 are equal to the contents of IAC5, also ANDed with the contents of IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 10 Inclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC5 and less than the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used. 11 Exclusive address range compare. IAC5 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC5 or is greater than or equal to the value specified in IAC6. IAC6 debug events do not occur. IAC5US and IAC5ER settings are used.
10:15	—	Reserved
16:17	IAC7US	Instruction Address Compare 7 User/Supervisor Mode 00 IAC7 debug events not affected by MSR_{PR} 01 Reserved

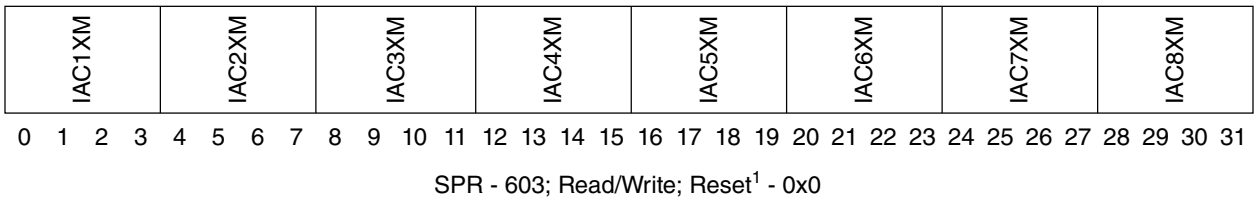
Table continues on the next page...

Table 65-10. DBCR5 field descriptions (continued)

Bit(s)	Name	Description
		10 IAC7 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 IAC7 debug events can only occur if MSR_{PR}=1 (User mode)
18:19	IAC7ER	Instruction Address Compare 7 Effective/Real Mode 00 IAC7 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC7 debug events are based on effective address and can only occur if MSR_{IS}=1
20:21	IAC8US	Instruction Address Compare 8 User/Supervisor Mode 00 IAC8 debug events not affected by MSR_{PR} 01 Reserved 10 IAC8 debug events can only occur if MSR_{PR}=0 (Supervisor mode). 11 IAC8 debug events can only occur if MSR_{PR}=1. (User mode)
22:23	IAC8ER	Instruction Address Compare 8 Effective/Real Mode 00 IAC8 debug events are based on effective address 01 Unimplemented (Book E real address compare), no match can occur 10 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=0 11 IAC8 debug events are based on effective address and can only occur if MSR_{IS}=1
24:25	IAC78M	Instruction Address Compare 7/8 Mode 00 Exact address compare. IAC7 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC7. IAC8 debug events can only occur if the address of the instruction fetch is equal to the value specified in IAC8. 01 Address bit match. IAC7 debug events can occur only if the address of the instruction fetch, ANDed with the contents of IAC8 are equal to the contents of IAC7, also ANDed with the contents of IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 10 Inclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is greater than or equal to the value specified in IAC7 and less than the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used. 11 Exclusive address range compare. IAC7 debug events can occur only if the address of the instruction fetch is less than the value specified in IAC7 or is greater than or equal to the value specified in IAC8. IAC8 debug events do not occur. IAC7US and IAC7ER settings are used.
26:31	—	Reserved

65.3.2.6 Debug Control Register 6 (DBCR6)

Debug Control Register 6 is used to extend instruction address compare matching functionality. DBCR6 is shown in the following figure.



SPR - 603; Read/Write; Reset[1] - 0x0

Figure 65-8. DBCR6 Register

Note

[1] DBCR6 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 6.

Table 65-11. DBCR6 field descriptions

Bit	Name	Description
0:3	IAC1XM	Instruction Address Compare 1 Extended Mask Control. IAC1XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC1 when comparing the storage address with the value in IAC1 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
4:7	IAC2XM	Instruction Address Compare 2 Extended Mask Control. IAC2XM allows for binary power of 2 address range compares for IAC2 without requiring the use of IAC1. 0000 No additional masking when DBCR1[IAC12M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC2 when comparing the storage address with the value in IAC2 for exact address compare (DBCR1[IAC12M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
8:11	IAC3XM	Instruction Address Compare 3 Extended Mask Control. IAC3XM allows for binary power of 2 address range compares for IAC1 without requiring the use of IAC2. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC3 when comparing the storage address with the value in IAC3 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
12:15	IAC4XM	Instruction Address Compare 4 Extended Mask Controll. IAC4XM allows for binary power of 2 address range compares for IAC4 without requiring the use of IAC3. 0000 No additional masking when DBCR1[IAC34M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC4 when comparing the storage address with the value in IAC4 for exact address compare (DBCR1[IAC34M]=00). Ranges up to 4 KB are supported.

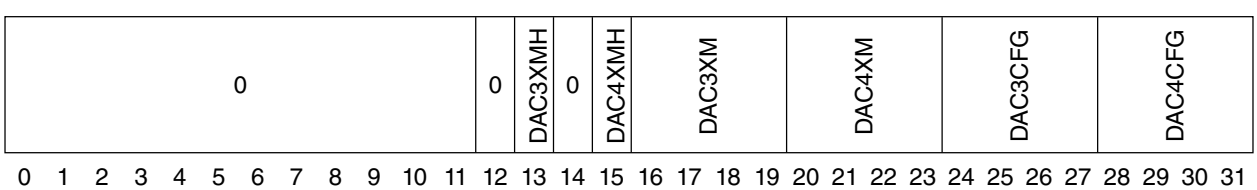
Table continues on the next page...

Table 65-11. DBCR6 field descriptions (continued)

Bit	Name	Description
		1101 - 1111 Reserved
16:19	IAC5XM	Instruction Address Compare 5 Extended Mask Control. IAC5XM allows for binary power of 2 address range compares for IAC5 without requiring the use of IAC6. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC5 when comparing the storage address with the value in IAC5 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
20:23	IAC6XM	Instruction Address Compare 6 Extended Mask Control. IAC6XM allows for binary power of 2 address range compares for IAC6 without requiring the use of IAC5. 0000 No additional masking when DBCR5[IAC56M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC6 when comparing the storage address with the value in IAC6 for exact address compare (DBCR5[IAC56M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
24:27	IAC7XM	Instruction Address Compare 7 Extended Mask Control. IAC7XM allows for binary power of 2 address range compares for IAC7 without requiring the use of IAC8. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC7 when comparing the storage address with the value in IAC7 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved
28:31	IAC8XM	Instruction Address Compare 8 Extended Mask Control. IAC8XM allows for binary power of 2 address range compares for IAC8 without requiring the use of IAC7. 0000 No additional masking when DBCR5[IAC78M]=00 0001 - 1100 Exact Match Bit Mask. Number of low order bits masked in IAC8 when comparing the storage address with the value in IAC8 for exact address compare (DBCR5[IAC78M]=00). Ranges up to 4 KB are supported. 1101 - 1111 Reserved

65.3.2.7 Debug Control Register 7 (DBCR7)

Debug Control Register 7 is used to enable and configure Data Address Compare 3 and 4 functionality. DBCR7 is shown in the following figure.



SPR - 596; Read/Write; Reset[1] - 0x0

Figure 65-9. DBCR7 Register

Note

[1] DBCR7 is reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 7.

Table 65-12. DBCR7 Field Descriptions

Bit	Name	Description
0:12	—	Reserved
13	DAC3XMH	Data Address Compare 3 Extended Mask Control High .DAC3XMH extends the range of the DAC3XM field 0 DAC3XM masks 0–15 low-order address bits 1 DAC3XM masks 16–31 low-order address bits
14	—	Reserved
15	DAC4XMH	Data Address Compare 4 Extended Mask Control High. . DAC4XMH extends the range of the DAC4XM field. 0 DAC4XM masks 0–15 low-order address bits 1 DAC4XM masks 16–31 low-order address bits
16:19	DAC3XM	Data Address Compare 3 Extended Mask Control. DAC3XM allows for binary power of 2 address range compares for DAC3 without requiring the use of DAC4. Value of DAC3XMH || DAC3XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC3 when comparing the storage address with the value in DAC3 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
20:23	DAC4XM	Data Address Compare 4 Extended Mask Control. DAC4XM allows for binary power of 2 address range compares for DAC4 without requiring the use of DAC3. Value of DAC4XMH || DAC4XM: 00000 No additional masking when DBCR8[DAC34M] = 00 00001 - 11111 Exact Match Bit Mask. One to 31 low-order bits are masked in DAC4 when comparing the storage address with the value in DAC4 for exact address compare (DBCR8[DAC34M] = 00). Address ranges of 2 bytes to 2GB are supported.
24:27	DAC3CFG	Data Address Compare 3 Configuration. DAC3CFG controls whether DAC3 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, a watchpoint-only function, or a stack limit checking function (with watchpoint). 0000 DAC3 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC3 debug watchpoints are enabled only for store-type data storage accesses

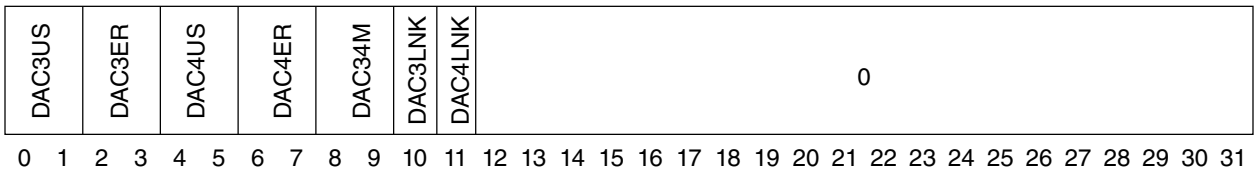
Table continues on the next page...

Table 65-12. DBCR7 Field Descriptions (continued)

Bit	Name	Description
		0010 DAC3 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100Reserved 0101 DAC3 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC3 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC3 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1000 DAC3 address comparisons are used for stack limit checking. DAC3 address comparisons are qualified with use of GPR R1 in <EA> calculation for most load or store instructions. No debug events occur for DAC3. When a qualified match occurs, a machine check exception is generated for supervisor-mode accesses or a DSI is generated for user-mode accesses, and a DAC3 watchpoints is generated. 1001 - 1111 Reserved
28:31	DAC4CFG	Data Address Compare 4 Configuration. DAC4CFG controls whether DAC4 data address comparisons utilize the normal compare operation for generating both debug events and watchpoints, or a watchpoint-only function. 0000 DAC4 debug watchpoints are enabled for load-type or store-type data storage accesses 0001 DAC4 debug watchpoints are enabled only for store-type data storage accesses 0010 DAC4 debug watchpoints are enabled only for load-type data storage accesses 0011 Reserved 0100 Reserved 0101 DAC4 debug events and watchpoints are enabled only for store-type data storage accesses 0110 DAC4 debug events and watchpoints are enabled only for load-type data storage accesses 0111 DAC4 debug events and watchpoints are enabled for load-type or store-type data storage accesses 1xxx Reserved

65.3.2.8 Debug Control Register 8 (DBCR8)

Debug Control Register 8 is used to configure Data Address Compare 3 and 4 operation. The DBCR8 register is shown in the following figure.



SPR - 597; Read/Write; Reset[1] - 0x0

Figure 65-10. DBCR8 Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for Debug Control Register 8.

Table 65-13. DBCR8 Bit Definitions

Bit(s)	Name	Description
0:1	DAC3US	Data Address Compare 3 User/Supervisor Mode 00 - DAC3 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC3 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC3 debug events can only occur if MSR_{PR}=1. (User mode)
2:3	DAC3ER	Data Address Compare 3 Effective/Real Mode 00 - DAC3 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=0 11 - DAC3 debug events are based on effective address and can only occur if MSR_{DS}=1
4:5	DAC4US	Data Address Compare 4 User/Supervisor Mode. 00 - DAC4 debug events not affected by MSR_{PR} 01 - Reserved 10 - DAC4 debug events can only occur if MSR_{PR}=0 (Supervisor mode) 11 - DAC4 debug events can only occur if MSR_{PR}=1. (User mode)
6:7	DAC4ER	Data Address Compare 4 Effective/Real Mode 00 - DAC4 debug events are based on effective address 01 - Unimplemented (Book E real address compare), no match can occur 10 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=0

Table continues on the next page...

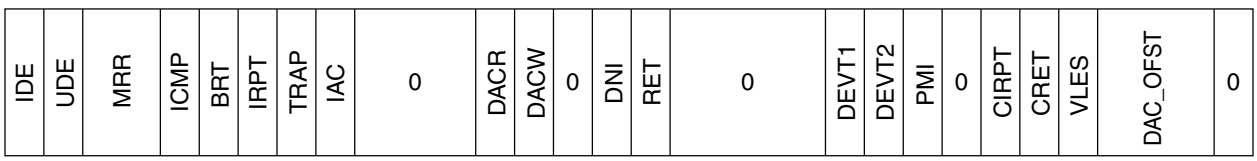
Table 65-13. DBCR8 Bit Definitions (continued)

Bit(s)	Name	Description
		11 - DAC4 debug events are based on effective address and can only occur if MSR_{DS}=1
8:9	DAC34M	Data Address Compare 3/4 Mode 00 - Exact address compare. DAC3 debug events can only occur if the address of the data access is equal to the value specified in DAC3. DAC4 debug events can only occur if the address of the data access is equal to the value specified in DAC4. 01 - Address bit match. DAC3 debug events can occur only if the address of the data access ANDed with the contents of DAC4, are equal to the contents of DAC3 also ANDed with the contents of DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 10 - Inclusive address range compare. DAC3 debug events can occur only if the address of the data access is greater than or equal to the value specified in DAC3 and less than the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used. 11 - Exclusive address range compare. DAC3 debug events can occur only if the address of the data access is less than the value specified in DAC3 or is greater than or equal to the value specified in DAC4. DAC4 debug events do not occur. DAC3US and DAC3ER settings are used.
10	DAC3LNK	Data Address Compare 3 Linked 0 - No effect 1 - DAC3 debug events are linked to IAC5 debug events. IAC5 debug events do not affect DBSR When linked to IAC5, DAC3 debug events are conditioned based on whether the instruction also generated an IAC5 debug event. Note that linking is only available in EDM or IDM.
11	DAC4LNK	Data Address Compare 4 Linked 0 - No effect 1 - DAC4 debug events are linked to IAC7 debug events. IAC7 debug events do not affect DBSR When linked to IAC7, DAC4 debug events are conditioned based on whether the instruction also generated an IAC7 debug event. Note that linking is only available in EDM or IDM.
12:31	—	Reserved

65.3.2.9 Debug Status Register (DBSR)

The Debug Status Register (DBSR) contains status on debug events and the most recent processor reset. The Debug Status Register is set via hardware, and read and cleared via software. Bits in the Debug Status Register can be cleared using **mtspr***DBSR,RS*. Clearing is done by writing to the Debug Status Register with a 1 in any bit position that is to be cleared and 0 in all other bit positions. The write data to the Debug Status Register is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no

effect. Debug Status bits are set by Debug events only while Internal Debug Mode is enabled ($DBCR0_{IDM}$=1). When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1, $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR other than MRR, DAC_OFST, or VLES will cause a debug interrupt to be generated. The debug interrupt handler is responsible for clearing DBSR bits prior to returning to normal execution. When resource sharing is enabled, ($EDBCR0_{EDM}$=1 and $EDBRAC0_{IDM}$=1), only software-owned resources may be modified by software, and status bits associated with hardware-owned resources will not be set by hardware in DBSR. The DBSR register is shown in the following figure.



SPR - 304; Read/Clear; Reset - 0x1000_0000

Figure 65-11. DBSR Register

The following table provides bit definitions for the Debug Status Register.

Table 65-14. DBSR Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if MSR_{DE}=0 and $DBCR0_{IDM}$=1 and a debug event causes its respective Debug Status Register bit to be set to 1. It may also be set to '1' if an imprecise debug event occurs due to a DAC event on a load or store which is terminated with error, or if an ICMP event occurs in conjunction with a EFPU FP round exception.
1	UDE	Unconditional Debug Event Set to 1 if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 - No reset occurred since these bits were last cleared by software 01 - A hard reset occurred since these bits were last cleared by software 10 - Reserved 11 - Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event

Table continues on the next page...

Table 65-14. DBSR Bit Definitions (continued)

Bit(s)	Name	Description
		Set to 1 if a Trap Taken debug event occurred.
8	IAC	Instruction Address Compare Debug Event Set to 1 if an IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a write-type DAC debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $PMGC0_{UDI}$=1
24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 if a Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if an ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $DBSR_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

65.3.2.10 Debug Data Effective Address Register (DDEAR)

The Debug Data Effective Address Register (DDEAR) contains address information for data address compare debug events (DAC). DDEAR is update by hardware with the effective address of the load, store, or cache control operation when a data address compare event is recorded in DBSR if the previous values of the $DBSR_{DAC\{R,W\}}$ bits are zero. Once a DDEAR update is performed, these bits must be cleared by software prior to another DDEAR hardware update occurring. A subsequent DAC event will not update the DDEAR register if either of the $DBSR_{DAC\{R,W\}}$ bits are set, in order to capture the first event address.

The DDEAR register is shown in the following table.

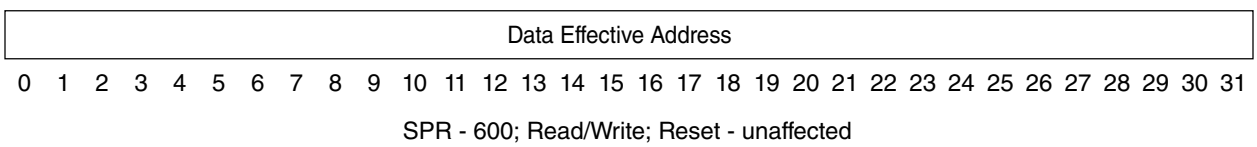


Figure 65-12. DDEAR Register

65.3.3 External Debug Resource Allocation Control Register (EDBRAC0)

The External Debug Resource Allocation Control Register (EDBRAC0) controls resource allocation when $EDBCR0_{EDM}$ is set to '1'. EDBRAC0 provides a mechanism for the hardware debugger to share certain debug resources with software. Individual resources are allocated based on the settings of EDBRAC0 when $EDBCR0_{EDM}$=1. EDBRAC0 settings are ignored when $EDBCR0_{EDM}$=0.

Hardware-owned resources which generate debug events update EDBSR0 instead of DBSR and cause entry into debug mode if the event is not masked in EDBSRMSK0, while software-owned resources which generate debug events if $DBCR0_{IDM}$=1 update DBSR, causing debug interrupts to occur if MSR_{DE}=1. EDBRAC0 is controlled via the OnCE port hardware, and is read-only to software.

The DBSR status register is always owned by software. Debug status bits in DBSR are set by software-owned debug events only while Internal Debug Mode is enabled. When debug interrupts are enabled (MSR_{DE}=1 $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=0, or MSR_{DE}=1, $DBCR0_{IDM}$=1 and $EDBCR0_{EDM}$=1 and software is allocated resource(s) via EDBRAC0), a set bit in DBSR by an event which is software-owned (other than MRR, DAC_OFST, or VLES) will cause a debug interrupt to be generated.

Debug status bits in EDBSR0 are set by hardware-owned debug events only while External Debug Mode is enabled ($EDBCR0_{EDM}$=1). When $EDBCR0_{EDM}$=1, a set bit in EDBSR0 by an event which is hardware-owned (other than IDE, DAC_OFST, or VLES) will cause entry into debug mode unless entry is masked via EDBSRMSK0.

If $EDBCR0_{EDM}$=1, DBSR status bits corresponding to hardware-owned debug events are masked from being set by hardware.

Software-owned resources may be modified by software, but only the corresponding control bits in DBCR0–8 are affected by execution of a **mtspr**, thus only a portion of these registers may be affected, depending on the allocation settings in EDBRAC0. The debug interrupt handler is still responsible for clearing DBSR bits for software-owned resources prior to returning to normal execution. Hardware always has full access to all registers and register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in EDBRAC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

The EDBRAC0 register is shown in the following figure.

0	IDM	RST	UDE	ICMP	BRT	IRPT	TRAP	IAC1	IAC2	IAC3	IAC4	DAC1	DAC34	DAC2	0	RET	IAC5	IAC6	IAC7	IAC8	DEVT1	DEVT2	PMI	0	CIRPT	CRET	DNI	DQM	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

SPR - 638; Read-only by Software; Reset - Unaffected by **p_reset_b,** reset to 0x00000180 by **m_por** or while in the test-logic-reset OnCE controller state

Figure 65-13. EDBRAC0 Register

Table 65-14 provides bit definitions for the Debug External Resource Control Register. Note that EDBRAC0 controls are disabled when EDBCR0EDM=0.

Table 65-15. EDBRAC0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	IDM	Internal Debug Mode control 0 - Internal Debug mode may not be enabled by software. $DBCR0_{IDM}$ is owned exclusively by hardware. **mtspr** DBCR0–8 and other debug registers is always ignored. No resource sharing occurs, regardless of the settings of other fields in EDBRAC0. Hardware exclusively owns all resources. 1 - Internal Debug mode may be enabled by software. $DBCR0_{IDM}$ is owned by software. $DBCR0_{IDM}$ is software readable/writable. When $EDBRAC0_{IDM}$=1, software writes to hardware-owned bits in DBCR0–8 via **mtspr** are ignored.

Table continues on the next page...

Table 65-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
2	RST	Reset Field Control 0 - $DBCR0_{RST}$ owned exclusively by hardware debug. No **mtspr** access by software to $DBCR0_{RST}$ field. 1 - $DBCR0_{RST}$ accessible by software debug. $DBCR0_{RST}$ is software readable/writable.
3	UDE	Unconditional Debug Event 0 - Event owned by hardware debug. 1 - Event owned by software debug.
4	ICMP	Instruction Complete Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{ICMP}$ field. 1 - Event owned by software debug. $DBCR0_{ICMP}$ is software readable/writable.
5	BRT	Branch Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{BRT}$ field. 1 - Event owned by software debug. $DBCR0_{BRT}$ is software readable/writable.
6	IRPT	Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{IRPT}$ field. 1 - Event owned by software debug. $DBCR0_{IRPT}$ is software readable/writable.
7	TRAP	Trap Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{TRAP}$ field. 1 - Event owned by software debug. $DBCR0_{TRA}$P is software readable/writable.
8	IAC1	Instruction Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC1 control and status fields. 1 - Event owned by software debug. IAC1 control fields are software readable/writable.
9	IAC2	Instruction Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC2 control and status fields. 1 - Event owned by software debug. IAC2 control fields are software readable/writable.
10	IAC3	Instruction Address Compare 3 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC3 control and status fields. 1 - Event owned by software debug. IAC3 control fields are software readable/writable.
11	IAC4	Instruction Address Compare 4 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC4 control and status fields. 1 - Event owned by software debug. IAC4 control fields are software readable/writable.
12	DAC1	Data Address Compare 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC1 control and status fields.

Table continues on the next page...

Table 65-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		1 - Event owned by software debug. DAC1 control fields are software readable/ writable.
13	DAC34	Data Address Compare 3 and 4 Debug Events 0 - Events owned by hardware debug. No **mtspr** access by software to DAC3 and DAC4 control and status fields. 1 - Event owned by software debug. DAC3 and DAC4 control fields are software readable/writable.
14	DAC2	Data Address Compare 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to DAC2 control and status fields. 1 - Event owned by software debug. DAC2 control fields are software readable/ writable.
15	—	Reserved
16	RET	Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{RET}$ field. 1 - Event owned by software debug. $DBCR0_{RET}$ is software readable/writable.
17	IAC5	Instruction Address Compare 5 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC5 control and status fields. 1 - Event owned by software debug. IAC5 control fields are software readable/writable.
18	IAC6	Instruction Address Compare 6 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC6 control and status fields. 1 - Event owned by software debug. IAC6 control fields are software readable/writable.
19	IAC7	Instruction Address Compare 7 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC7 control and status fields. 1 - Event owned by software debug. IAC7 control fields are software readable/writable.
20	IAC8	Instruction Address Compare 8 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to IAC8 control and status fields. 1 - Event owned by software debug. IAC8 control are software readable/writable.
21	DEVT1	External Debug Event Input 1 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT1}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT1}$ is software readable/writable.
22	DEVT2	External Debug Event Input 2 Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{DEVT2}$ field. 1 - Event owned by software debug. $DBCR0_{DEVT2}$ is software readable/writable.
23	PMI	Performance Monitor Interrupt Debug Event

Table continues on the next page...

Table 65-15. EDBRAC0 Bit Definitions (continued)

Bit(s)	Name	Description
		0 - Event owned by hardware debug. No **mtspr** access by software to the PMRs. Performance monitor interrupts set $EDBSR0_{PMI}$ regardless of the setting of $PMGC0_{UDI}$ 1 - Event owned by software debug. PMRs are software readable/writable. **Note:** this bit is reset to '1'.
24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CIRPT}$ field. 1 - Event owned by software debug. $DBCR0_{CIRPT}$ is software readable/writable.
26	CRET	Critical Return Debug Event 0 - Event owned by hardware debug. No **mtspr** access by software to $DBCR0_{CRET}$ field. 1 - Event owned by software debug. $DBCR0_{CRET}$ is software readable/writable.
27	DNI	DNI Instruction Debug Control 0 - DNI resource owned by hardware debug. Execution of an **e_dni** or **se_dni** instruction results in entry into debug mode. 1 - DNI resource owned by software debug. Execution of an **e_dni** or **se_dni** instruction results in either a debug interrupt ($DBCR0_{IDM}$=1 and MSR_{DE}=1) or a nop ($DBCR0_{IDM}$=0 or MSR_{DE}=0). **Note:** DNI events are not blocked during a debug session
28	DQM	Data Acquisition Messaging Registers 0 - $DEVENT_{DQTAG}$ and DDAM register are exclusively owned by hardware debug. No **mtspr** access by software to $DEVENTD_{QTAG}$ field or DDAM register. Attempted access by software is ignored. 1 - $DEVENT_{DQTAG}$ and DDAM register are owned by software. Software has read/write access to $DEVENT_{DQTAG}$ field and DDAM register.
29:31	—	Reserved

Table 65-16 shows which resources are controlled by EDBRAC0 settings.

Table 65-16. EDBRAC0 Resource Control

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$		$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	**Software Accessible via mtspr,affected by p_reset_b**
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	All debug registers
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IDM}$
1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{RST}$
1	1	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{UDE}$

Table continues on the next page...

Table 65-16. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$		$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
1	1	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{ICMP}$
1	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{BRT}$
1	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{IRPT}$
1	1	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR0_{TRAP}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC1, $DBCR0_{IAC1}$, $DBCR1_{IAC1US,IAC1ER}$, $DBCR6_{IAC1XM}$
1	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC2, $DBCR0_{IAC2}$, $DBCR1_{IAC2US,IAC2ER}$, $DBCR6_{IAC2XM}$
1	1	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC12M}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC3, $DBCR0_{IAC3}$, $DBCR1_{IAC3US,IAC3ER}$, $DBCR6_{IAC3XM}$
1	1	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC4, $DBCR0_{IAC4}$, $DBCR1_{IAC4US,IAC4ER}$, $DBCR6_{IAC4XM}$
1	1	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR1_{IAC34M}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC5, $DBCR0_{IAC5}$, $DBCR5_{IAC5US,IAC5ER}$, $DBCR6_{IAC5XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC6, $DBCR0_{IAC6}$, $DBCR5_{IAC6US,IAC6ER}$, $DBCR6_{IAC6XM}$
1	1	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC56M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	IAC7,

Table continues on the next page...

Table 65-16. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$		$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	**Software Accessible via mtspr,affected by p_reset_b**
																												$DBCR0_{IAC7}$, $DBCR5_{IAC7US,IAC7ER}$, $DBCR6_{IAC7XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	IAC8, $DBCR0_{IAC8}$, $DBCR5_{IAC8US,IAC8ER}$, $DBCR6_{IAC8XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	-	-	-	-	-	-	-	$DBCR5_{IAC78M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	DAC1, DVC1, DVC1U $DBCR0_{DAC1}$, $DBCR2_{DAC1US,DAC1ER}$, $DBCR2_{DVC1M,DVC1BE}$ $DBCR4_{DVC1C,DAC1XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	DAC3, DAC4 $DBCR7_{DAC\{3,4\},DAC\{3,4\}CFG, DAC\{3,4\}XM, DAC\{3,4\}XMH}$ $DBCR8_{DAC\{3,4\}US, DAC\{3,4\}ER, DAC\{3,4\}M}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	DAC2, DVC2, DVC2U $DBCR0_{DAC2}$, $DBCR2_{DAC2US,DAC2ER}$, $DBCR2_{DVC2M,DVC2BE}$ $DBCR4_{DVC2C,DAC2XM}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	1	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC12M}$
1	1	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC1LNK}$
1	1	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	$DBCR2_{DAC2LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	$DBCR8_{DAC3LNK}$

Table continues on the next page...

Table 65-16. EDBRAC0 Resource Control (continued)

$EDBCR0_{EDM}$	$EDBRAC0_{IDM}$	$EDBRAC0_{RST}$	$EDBRAC0_{UDE}$	$EDBRAC0_{ICMP}$	$EDBRAC0_{BRT}$	$EDBRAC0_{IRPT}$	$EDBRAC0_{TRAP}$	$EDBRAC0_{IAC1}$	$EDBRAC0_{IAC2}$	$EDBRAC0_{IAC3}$	$EDBRAC0_{IAC4}$	$EDBRAC0_{IAC5}$	$EDBRAC0_{IAC6}$	$EDBRAC0_{IAC7}$	$EDBRAC0_{IAC8}$	$EDBRAC0_{DAC1}$	$EDBRAC0_{DAC34}$	$EDBRAC0_{DAC2}$	$EDBRAC0_{RET}$	$EDBRAC0_{DEVT1}$	$EDBRAC0_{DEVT2}$	$EDBRAC0_{PMI}$		$EDBRAC0_{CIRT}$	$EDBRAC0_{CRET}$	$EDBRAC0_{DNI}$	$EDBRAC0_{DQM}$	Software Accessible via mtspr,affected by p_reset_b
1	1	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	1	-	-	-	-	-	-	-	-	-	-	$DBCR8_{DAC4LNK}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	$DBCR0_{RET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	$DBCR0_{DEVT1}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	$DBCR0_{DEVT2}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	All PMRs
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	$DBCR0_{CIRPT}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	$DBCR0_{CRET}$
1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		-	-	-	-	-	-	1	-	
1	-[1]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	$DEVENT_{DQTAG}$, DDAM

1. Note: IDM not required to be set to enable software access.

EDBRAC0 also controls which bits or fields in DBCR0–8 and the PMRs are reset by assertion of **p_reset_b** when $EDBCR0_{EDM}$=1. Only software-owned bits or fields as shown in Table 65-16 are affected in this case, except that $DBCR0_{RST}$ and $DBSR_{MRR}$ are updated by assertion of **p_reset_b** regardless of the value of $EDBCR0_{EDM}$ or EDBRAC0.

65.3.4 Debug Event Select Register (DEVENT)

The Debug Event Select Register allows instrumented software to internally generate signals when a **mtspr** instruction is executed and this register is accessed. The values written to this register determine which of the **p_devnt_out[0:7]** processor output signals are asserted upon access. Writing a '1' to any of these bit positions will cause a one clock pulse to be generated on the corresponding output. For **p_devnt_out[0:3]**, a corresponding **jd_watchpt[x]** output is asserted as well to indicate a watchpoint has occurred. These signals may be used for internal core debug resources as well as for SoC level cross-triggering. See the SoC User's Manual for more information on SoC use cases.

The $DEVENT_{DEVNT}$ register field value is undefined on a read; it may or may not remain set to the last value written. Since it is unconditionally shared by hardware debug and software, software should not rely on any value remaining.

The upper 8 bits of the DEVENT register also provide the DQTAG used to identify channels within Data Acquisition Messages. See the "Data Acquisition ID Tag Field" section in the Core (e200z4204n3) Nexus 3 Module chapter for more detail on the DQTAG.

The DEVENT register is shown in the following figure.

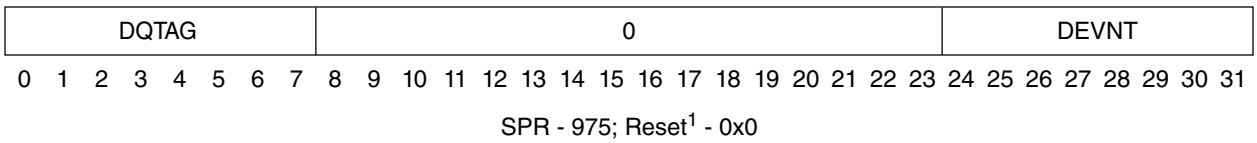


Figure 65-14. DEVENT Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**. Note that DEVNT field is shared by hardware and software but is always reset by **p_reset_b**.

The following table provides bit definitions for the Debug Event Register.

Table 65-17. DEVENT Bit Definitions

Bit(s)	Name	Description
0:7	DQTAG	Data Acquisition Message IDTAG channel identifier (supplied to Nexus 3)
8:23	—	Reserved, should be cleared.
24:31	DEVNT	Debug Event Signals 00000000 - No signal is asserted xxxxxxx1 - **p_devnt_out[0]** and **jd_watchpt[12]** are asserted for one clock xxxxxx1x - **p_devnt_out[1]** and **jd_watchpt[13]** are asserted for one clock xxxxx1xx - **p_devnt_out[2]** and **jd_watchpt[20]** are asserted for one clock xxxx1xxx - **p_devnt_out[3]** and **jd_watchpt[21]** are asserted for one clock xxx1xxxx - **p_devnt_out[4]** is asserted for one clock xx1xxxxx - **p_devnt_out[5]** is asserted for one clock x1xxxxxx - **p_devnt_out[6]** is asserted for one clock 1xxxxxxx - **p_devnt_out[7]** is asserted for one clock

65.3.5 Debug Data Acquisition Message Register (DDAM)

The Debug Data Acquisition Message Register allows instrumented software to generate real-time Data Acquisition Messages (as defined by Nexus 3) via a **mtspr** instruction to this register. See the "Data Acquisition Messaging" section in the Core (e200z4204n3) Nexus 3 Module chapter for details.

The DDAM register is shown in the following figure.

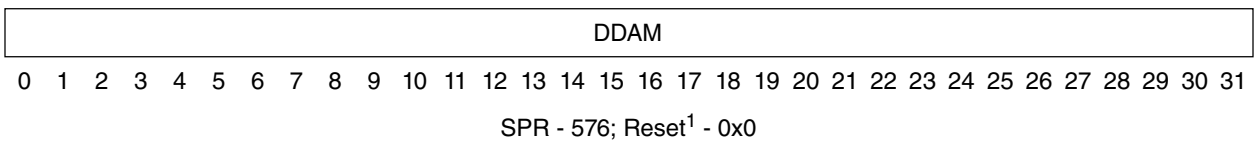


Figure 65-15. DDAM Register

Note

[1] Reset by processor reset **p_reset_b** if $EDBCR0_{EDM}$=0, as well as unconditionally by **m_por**. If $EDBCR0_{EDM}$=1, EDBRAC0 masks off hardware-owned resources from reset by **p_reset_b** and only software-owned resources indicated by EDBRAC0 will be reset by **p_reset_b**.

The following table provides bit definitions for the Debug Data Acquisition Message Register.

Table 65-18. DDAM Bit Definitions

Bit(s)	Name	Description
0:31	DDAM	Value to be transmitted in a Data Acquisition Message (DQM) (supplied to Nexus 3 with strobe)

65.4 Using Debug Resources for Stack Limit Checking

The DAC1,2 and DAC3,4 resources can be used for stack overflow/underflow detection when not being used as a hardware or software debug resource. Stack limit checking is available regardless of EDM or IDM mode, and when resources used for stack limit checking are owned by software, will utilize a DSI or machine check exception. Software-owned stack limit checking does not require IDM to be set. Hardware owned stack limit checking requires EDM to be set.

When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by software, DAC events are not generated for resources configured to perform stack limit checking, and no DBSR DAC status flag will be set due to a detected

stack limit violation. Instead, depending on the processor mode, a data storage interrupt or a machine check exception is signaled. When stack limit checking is enabled, and DAC resources used for stack limit checking are owned by hardware, DAC events will be generated for resources configured to perform stack limit checking, and the EDBSR0 DAC status flag will be set due to a detected stack limit violation, causing entry into debug halted mode in the same way as a DAC exception normally does. The only difference is that qualification of the access address is performed as discussed in the next paragraph.

Stack limit checking is implemented in the same way as range compares using DAC1,2 or 3,4, or extended masking using DAC1 or DAC3, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DAC resources configured to perform stack limit checking are not owned by hardware, if a stack limit violation occurs when performing the load or store, the access is aborted, and an error report machine check is generated, with MCSRR0 pointing to the address of the load or store access which generated the stack overflow/underflow. If DAC resources configured to perform stack limit checking are owned by hardware, then a normal DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event.

Enabling of this functionality is described in more detail in Table 65-9. Note that IDM is not required to be set for software-owned stack limit checking.

In contrast to the normal case of DAC address matching resulting in a debug event, the stack limit check address compare logic operates differently for stack limit checking. When using resources for stack limit checking, a DACn hit to a resource enabled for performing stack limit checking on a stack access indicates a stack limit violation.

When stack limit checking is enabled by the setting of the $DBCR4_{DAC1CFG}$ field to '1000' or the $DBCR8_{DAC3CFG}$ field to '1000' (or both), and the corresponding DACn resources are owned by software, DACn events are not generated and the DBSR DAC status flag will not be set due to a detected stack limit hit. Instead, if stack limit checking is enabled for supervisor mode stack accesses in DAC1 or DAC3, and a compare hit occurs for a supervisor mode stack access (A load or store using GPR R1 in the <EA> calculation), a machine check exception is signaled. If stack limit checking is enabled for user mode accesses, a DSI exception is signaled when a stack limit checking enabled DAC1 or DAC3 compare hit occurs for a user mode access. A watchpoint for DAC1 or DAC3 will be generated when the stack limit violation is detected, even though the instruction does not complete. Stack limit checking for supervisor mode stack accesses is considered

enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '10', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '10', or both. Stack limit checking for user mode stack accesses is considered enabled when either the $DBCR4_{DAC1CFG}$ field is set to '1000' with $DBCR2_{DAC1US}$ set to '00' or '01', or the $DBCR7_{DAC3CFG}$ field is set to '1000' with $DBCR8_{DAC3US}$ set to '00' or '01', or both. Note that unlike regular debug DAC events, both halves of a misaligned access are checked for limit violations.

When stack limit checking is enabled for a stack access, and DACn resources are owned by hardware, the EDBSR0 DAC status flag will be set due to a detected stack limit violation, to cause entry into debug halted mode or to generate a watchpoint, or both, in the same way as a DAC event normally does, i.e. after the access has completed. The only difference is that qualification of the access address is performed as discussed in the next paragraph. If the access is aborted due to a DSI or other exception such as machine check condition, the $EDBSR0_{IDE}$ status bit will also be set to indicate that the data access instruction was not completed.

Stack limit checking is implemented in the same way as address compares using DACn, but qualify a load or store access address with the use of GPR R1 as the base or index register used to compute an effective address when a load or store instruction is executed. No stack limit checking is performed for other instructions which may calculate an EA using GPR R1 such as cache, management instructions, or instructions which indicate GPR R1 is used to hold a decoration value (for decorated load/store type instructions). When DACn resources are not owned by hardware, if a stack limit violation occurs (a hit to a stack limit enabled DAC is detected) when performing the load or store, the access request is aborted and the access is not performed. Instead, for supervisor mode accesses, an error report machine check exception is generated, with MCSRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow, and for user mode accesses, a DSI exception is generated, with SRR0 pointing to the address of the load or store instruction which generated the stack overflow/underflow. If all stack limit check enabled DACn resources are owned by hardware, then a DAC event is generated (but qualified with use of GPR R1), and debug mode entry will occur in the same manner as for a non-stack limit DAC event. In this case, the instruction and access will normally have completed, in the same manner as a normal DAC event.

Independent limit checks for supervisor and user accesses may be implemented by allocating independent DACn resources to each, or a single limit may be applied using a single DACn resource. Typically, a DAC1,2 pair operating in exclusive address range mode is utilized for stack limit checking for a single shared limit. For separate limits, DAC3,4 may also be utilized. If more than one DACn resource is utilized, a DAC hit on any resource utilized for stack limit checking will cause the corresponding stack limit exception action to occur. If both a hardware-owned and a software-owned resource

generate a stack limit exception for a given load or store, the software resource will have priority, since it is detected prior to completion of the access, and the access is aborted, thus the hardware event will not occur.

Enabling of this functionality is described in more detail in the description of the DACnCFG fields in Table 65-9 and Table 65-12 in Debug Control and Status registers. Note that for DAC1 and DAC2, access type (read, write) control is part of DBCR0.

65.5 External Debug Support

External debug support is supplied through the OnCE controller serial interface which allows access to internal CPU registers and other system state while the CPU is halted in debug mode. All debug resources including DBCR0–8, DBSR, IAC1–8, DAC1–4, and DVC1–2 are accessible through the serial OnCE interface in external debug mode. Setting the $EDBCR0_{EDM}$/$DBCR0_{EDM}$ bit to '1' through the OnCE interface enables external debug mode, and unless otherwise permitted by the settings in EDBRAC0, disables software updates to the debug control registers. When $EDBCR0_{EDM}$ is set, debug events enabled to set respective status bits will also cause the CPU to enter Debug Mode if the event is not masked in EDBSRMSK0, as opposed to generating Debug Interrupts, unless the specific events are allocated to software via the settings in EDBRAC0. In Debug Mode, the CPU is halted at a recoverable boundary, and an external Debug Control Module may control CPU operation through the On-Chip Emulation logic (OnCE).

Note that the descriptions of events in the subsections of Software Debug Events and Exceptions refer to setting DBSR status bits, however, when resources are owned by hardware, the events for those resources set the respective status bits in EDBSR0 instead of DBSR.

Note

On the initial setting of $EDBCR0_{EDM}$ to '1', other bits in DBCR0 will remain unchanged. After $EDBCR0_{EDM}$ has been set, all debug register resources may be subsequently controlled through the OnCE interface. The CPU should be placed into debug mode via the OCR_{DR} control bit prior to writing EDM to '1'. This gives the debugger the opportunity to cleanly write to the DBCRx registers and the DBSR to clear out any residual state / control information which could cause unintended operation.

Note

It is intended for the CPU to remain in external debug mode ($EDBCR0_{EDM}$=1) in order to single step or perform other debug mode entry/ reentry via the OCR_{DR}, by performing go+noexit commands, or by assertion of the **jd_de_b** signal.

Note

$EDBCR0_{EDM}$ operation will be blocked if OnCE operation is disabled (**jd_en_once** negated) regardless of whether it is set or cleared. This means that if $EDBCR0_{EDM}$ was previously set, and then **jd_en_once** is negated (this should not occur), entry into debug mode will be blocked, and all *hardware* debug events are blocked. Watchpoints are not blocked.

Due to clock domain design, the CPU clock (**m_clk**) must be active in order to perform writes to debug registers other than the OnCE Command register (OCMD), the OnCE Control register (OCR), External Debug Control Register 0 (EDBCR0), External Debug Status register 0 (EDBSR0), External Debug Status Register Mask 0 (EDBSRMSK0), or the $EDBCR0_{EDM}$ bit. Register read data is synchronized back to the **j_tclk** clock domain. The OnCE Control register provides the capability of signaling the system level clock controller that the CPU clock should be activated if not already active.

Updates to the DBCRx, DBSR, and most other debug registers via the OnCE interface should be performed with the CPU in debug mode to guarantee proper operation. Due to the various points in the CPU pipeline where control is sampled and event handshaking is performed, it is possible that modifications to these registers while the CPU is running may result in early or late entry into debug mode, and may have incorrect status posted in the DBSR register.

If resource sharing is enabled via EDBRAC0, updates to the EDBRAC0, DBCRx, and DBSR registers must be performed with the CPU in debug mode, since simultaneous updates of register portions could otherwise be attempted, and such updates are not guaranteed to properly occur. The results of such an attempt are undefined.

65.5.1 External Debug Registers

The external debug registers are used for controlling several debug aspects of the core and reporting status while in External Debug Mode.

65.5.1.1 External Debug Control Register 0 (EDBCR0)

EDBCR0 is a control register accessible to an external debugger through the OnCE/Jtag port. An external development tool can write to this register in order to enable external debug mode or to enable Debugger Notify Halt instructions (**e_dnh**, **se_dnh**), as well as to control aspects of Debugger Notify Interrupt instructions (**e_dni**, **se_dni**).

EDBCR0 is not accessible by software, However, the state of $EDBCR0_{EDM}$ is reflected as a read-only bit in $DBCR0_{EDM}$ to software. There is only one physical EDM bit implemented; it is reflected in both the DBCR0 and EDBCR0 registers, and may be written and read using either register by the hardware debugger. For future compatibility, EDBCR0 updates are preferred.

EDBCR0 is shown in the following figure.

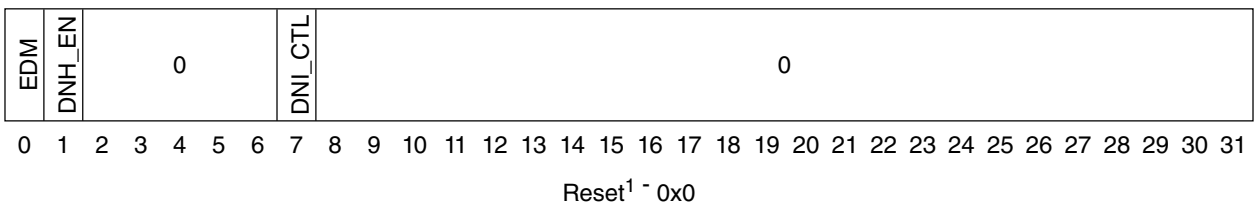


Figure 65-16. EDBCR0 Register

Note

[1] EDBCRO is affected (reset) by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state, but is not affected by **p_reset_b.**

The following table provides bit definitions for External Debug Control Register 0.

Table 65-19. EDBCR0 Bit Definitions

Bit(s)	Name	Description
0	EDM	External Debug Mode. This bit is also reflected in DBCR0 0 - External debug mode disabled. Internal debug events not mapped into external debug events. 1 - External debug mode enabled. Hardware-owned events will not cause the CPU to vector to interrupt code. Software is not permitted to write to debug registers {DBCRx, IAC1-8, DAC1-4, DVC1-2[U]} unless permitted by settings in EDBRAC0. When external debug mode is enabled, hardware-owned resources in debug registers are not affected by processor reset **p_reset_b**. This allows the debugger to set up hardware debug events which remain active across a processor reset.
1	DNH_EN	**dnh** Instruction Enable 0 - execution of **e_dnh** and **se_dnh** instructions cause illegal instruction exceptions to occur.

Table continues on the next page...

Table 65-19. EDBCR0 Bit Definitions (continued)

Bit(s)	Name	Description
		1 - execution of **e_dnh** and **se_dnh** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of EDM.
2:6	---	Reserved
7	DNI_CTL	**dni** Instruction Control 0 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is cared, and when MSRDE=0, execution of **e_dni** and **se_dni** instructions are nop'ed and no entry into debug mode occurs. 1 - When the **dni** resource is owned by hardware, the MSR_{DE} bit is don't-cared, and execution of **e_dni** and **se_dni** instructions cause entry into debug mode and a debug halt occurs, regardless of the value of MSR_{DE}. Note that this control bit is only used when the dni resource is owned by hardware via control in $EDBRAC0_{DNI}$, and thus also only when EDM=1
8:31	---	Reserved

65.5.1.2 External Debug Status Register 0 (EDBSR0)

The External Debug Status Register 0 (EDBSR0) contains status on debug events owned by hardware. The External Debug Status Register 0 is set via hardware, and read and cleared via OnCE access by the debugger. Clearing is done by writing to the External Debug Status Register via the OnCE port, with a '1' in any bit position that is to be cleared and '0' in all other bit positions. The write data to EDBSR0 is not direct data, but a mask. A '1' causes the bit to be cleared, and a '0' has no effect. The EDBSR0 register is shown in the following figure.

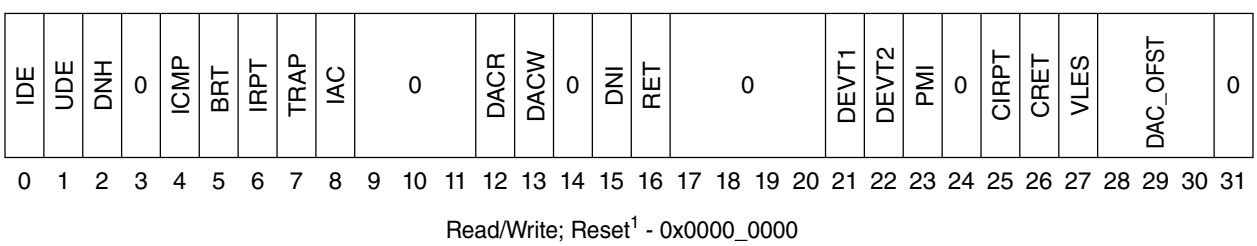


Figure 65-17. EDBSR0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=0.

The following table provides bit definitions for External Debug Status Register 0.

Table 65-20. EDBSR0 Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to 1 if $EDBCR0_{EDM}$=1 and an imprecise debug event occurs for a hardware-owned DAC event due to a load or store which is terminated with error, or if a hardware-owned ICMP event occurs in conjunction with a EFPU round exception. This bit will not be set for imprecise debug events which are masked via settings in EDBSRMSK0.
1	UDE	Unconditional Debug Event Set to 1 if a hardware-owned Unconditional debug event occurred.
2	DNH	Debugger Notify Halt Event Set to 1 if a debugger notify halt instruction was executed and caused a debug halt.
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 if a hardware-owned Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to 1 if a hardware-owned Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to 1 if a hardware-owned Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to 1 if a hardware-owned Trap Taken debug event occurred.
8	IAC	Instruction Address Compare 1 Debug Event Set to 1 if a hardware-owned IAC debug event occurred.
9:11	—	Reserved
12	DACR	Data Address Compare Read Debug Event Set to 1 if a hardware-owned read-type DAC debug event occurred
13	DACW	Data Address Compare Write Debug Event Set to 1 if a hardware-owned write-type DAC1 debug event occurred
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 if a hardware-owned DNI debug event occurred
16	RET	Return Debug Event Set to 1 if a hardware-owned Return debug event occurred
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 if a hardware-owned DEVT1 debug event occurred
22	DEVT2	External Debug Event 2 Debug Event Set to 1 if a hardware-owned DEVT2 debug event occurred
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 if a Performance Monitor Interrupt event occurred with $EDBRAC0_{PMI}$=0 regardless of the setting of $PMGC0_{UDI}$
24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event

Table continues on the next page...

Table 65-20. EDBSR0 Bit Definitions (continued)

Bit(s)	Name	Description
		Set to 1 if a hardware-owned Critical Interrupt Taken debug event occurred.
26	CRET	Critical Return Debug Event Set to 1 if a hardware-owned Critical Return debug event occurred
27	VLES	VLE Status Set to 1 if a hardware-owned ICMP, BRT, TRAP, RET, CRET, IAC, or DAC debug event occurred on a PowerISA VLE Instruction. Also set for execution of an e_dnh or se_dnh instruction when enabled by $EDBCR0_{DNH_EN}$. Undefined for IRPT, CIRPT, DEVT[1,2], and UDE events
28:30	DAC_OFST	Data Address Compare Offset Indicates offset-1 of saved DSRR0 value from the address of the load or store instruction which took a hardware-owned DAC Debug exception, unless a simultaneous DSI error occurs, in which case this field is set to 3'b000 and $EDBSR0_{IDE}$ is set to 1. Normally set to 3'b000 by a non-DVC DAC. A DVC DAC may set this field to any value.
31	—	Reserved

65.5.1.3 External Debug Status Register Mask 0 (EDBSRMSK0)

The External Debug Status Register Mask 0 (EDBSRMSK0) is used to mask debug events set in EDBSR0 from causing entry into debug halted mode. A '1' stored in any mask bit prevents debug mode entry caused by the corresponding bit being set in EDBSR0. The mask has no effect on DBSR actions or on the setting of EDBSR0 status bits by hardware-owned events, except that the IDE bit will not be set by imprecise hardware-owned debug events which are masked. EDBSRMSK0 may be used to allow debug events owned by hardware to be configured for watchpoint generation purposes without causing debug mode entry when the watchpoint occurs. EDBSRMSK0 is read and written via OnCE access by the debugger. No software access is provided. The EDBSRMSK0 register is shown in the following figure.

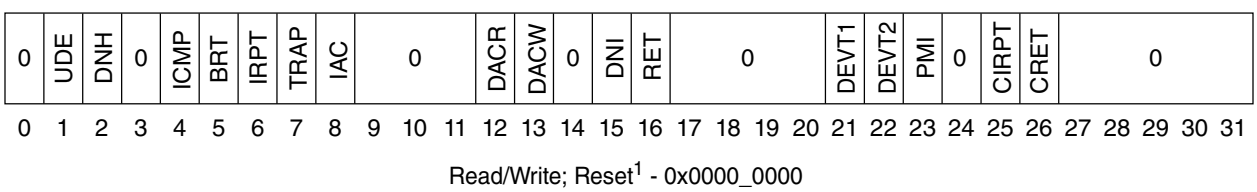


Figure 65-18. EDBSRMSK0 Register

Note

[1] Reset by **j_trst_b** or **m_por** assertion, and remains reset while in the Test_Logic_Reset state or while $EDBCRO_{EDM}$=0.

The following table provides bit definitions for External Debug Status Register Mask 0.

Table 65-21. EDBSRMSK0 Bit Definitions

Bit(s)	Name	Description
0	—	Reserved
1	UDE	Unconditional Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{UDE}$
2	DNH	Debugger Notify Halt Event Set to 1 to mask debug mode entry by $EDBSR0_{DNH}$
3	—	Reserved
4	ICMP	Instruction Complete Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{ICMP}$
5	BRT	Branch Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{BRT}$
6	IRPT	Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IRPT}$
7	TRAP	Trap Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{TRAP}$
8	IAC	Instruction Address Compare Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{IAC}$
9:11	—	Reserved
12	DACR	Data Address Compare 1 Read Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACR}$
13	DACW	Data Address Compare 1 Write Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DACW}$
14	—	Reserved
15	DNI	DNI Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DNI}$
16	RET	Return Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{RET}$
17:20	—	Reserved
21	DEVT1	External Debug Event 1 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT1}$
22	DEVT2	External Debug Event 2 Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{DEVT2}$
23	PMI	Performance Monitor Interrupt Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{PMI}$
24	—	Reserved
25	CIRPT	Critical Interrupt Taken Debug Event Set to 1 to mask debug mode entry by $EDBSR0_{CIRPT}$
26	CRET	Critical Return Debug Event

Table continues on the next page...

Table 65-21. EDBSRMSK0 Bit Definitions (continued)

Bit(s)	Name	Description
		Set to 1 to mask debug mode entry by $EDBSR0_{CRET}$
22:31	—	Reserved

65.5.1.4 External Debug Data Effective Address Register (EDDEAR)

The External Debug Data Effective Address Register (EDDEAR) contains address information for hardware-owned data address compare debug events. EDDEAR is update by hardware with the effective address of the load, store, or cache control operation when an unmasked data address compare event is recorded in EDBSR0 if the previous values of $EDBSR0_{DAC\{R,W\}}$ bits which are unmasked in EDBSRMSK0 are zero. Once an EDDEAR update is performed, these unmasked bits must be cleared by the hardware debugger prior to another EDDEAR hardware update occurring. A subsequent hardware-owned unmasked DAC event will not update the EDDEAR register if either of the $EDBSR0_{DAC\{R,W\}}$ bits are set and are not masked by EDBSRMSK0, in order to capture the first unmasked event address. EDDEAR is read and written via OnCE access by the debugger. No software access is provided.

The EDDEAR register is shown in the following figure.

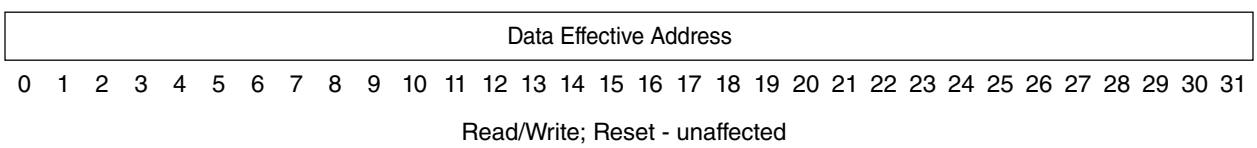


Figure 65-19. EDDEAR Register

65.5.2 OnCE Introduction

The e200z4204n3 on-chip emulation circuitry (OnCE™/Nexus Class 1 interface) provides a means of interacting with the e200z4204n3 core and integrated system so that a user may examine registers, memory, or on-chip peripherals facilitating hardware/ software development. OnCE operation is controlled via an industry standard IEEE 1149.1 TAP controller. By using public instructions, the external hardware debugger can freeze or halt the CPU, read and write internal state, and resume normal execution. The core does not contain IEEE 1149.1 standard boundary cells on its interface, as it is a building block for further integration. It does not support the JTAG related boundary scan instruction functionality, although JTAG public instructions may be decoded and signaled to external logic.

The OnCE logic provides for Nexus Class 1 static debug capability (utilizing the same set of resources available to software while in internal debug mode).

In order to enable full OnCE operation, the **jd_enable_once** input signal must be asserted. In some system integrations, this is automatic, since the input will be tied asserted. Other integrations may require the execution of the Enable OnCE command via the TAP and appropriate entry of serial data. Exact requirements will be documented by the integrated product specification. The **jd_enable_once** input signal should not change state during a debug session, or undefined activity may occur.

The following figures show the TAP controller state model and the TAP registers implemented by the OnCE logic.

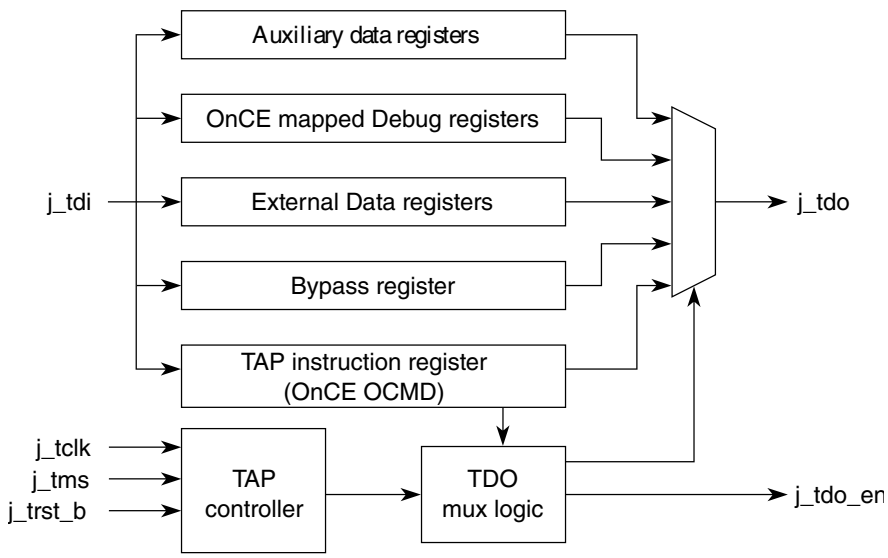
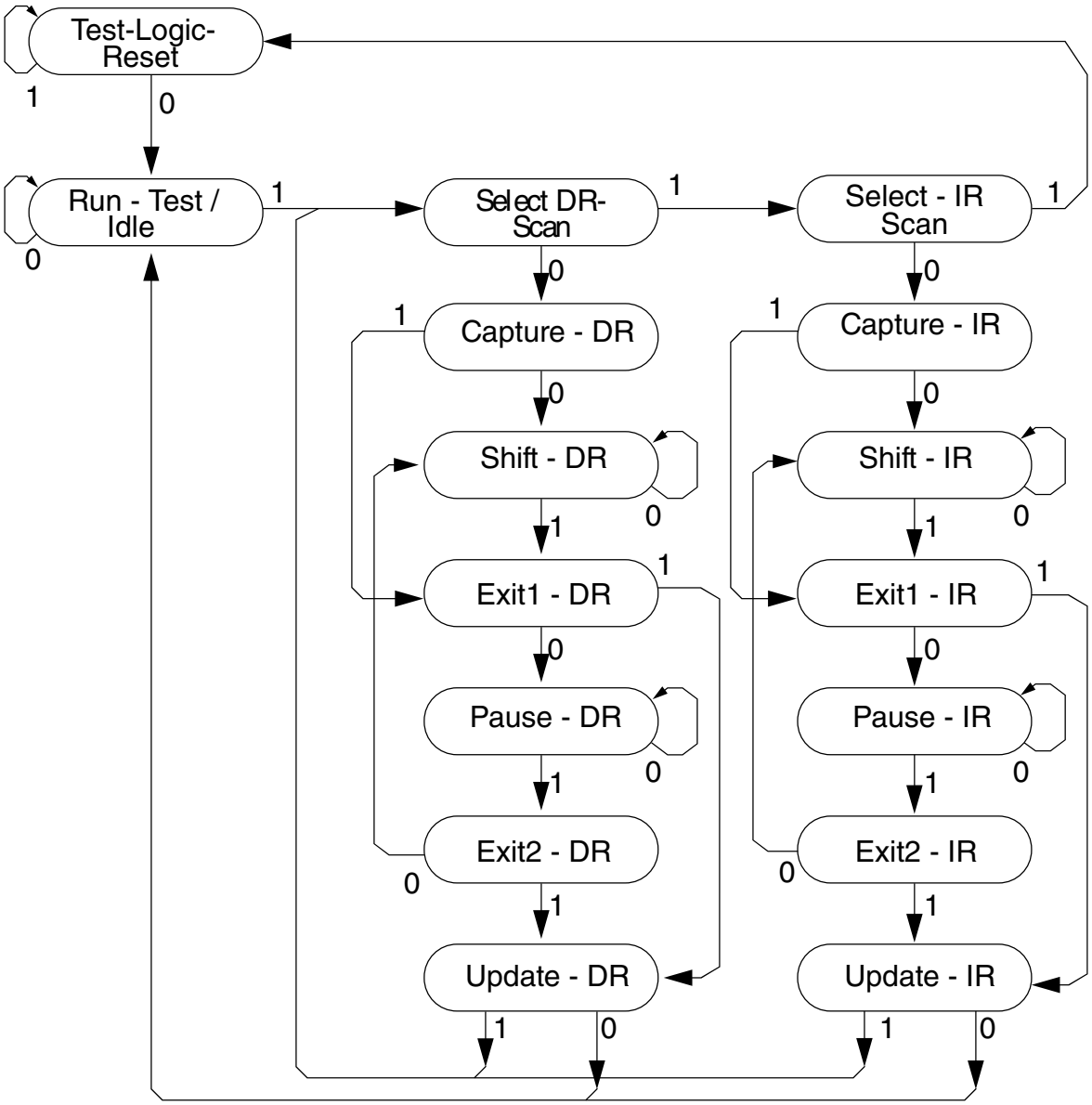


Figure 65-20. OnCE TAP Controller and Registers

The OnCE controller is implemented as a 16-state FSM, with a one-to-one correspondence to the states defined for the JTAG TAP controller.



Access to processor registers and the contents of memory locations are performed by enabling external debug mode (setting $EDBCR0_{EDM}$ to '1'), placing the processor into debug mode, followed by scanning instructions and data into and out of the CPU Scan Chain (CPUSCR); execution of scanned instructions by the CPU is used as the method to access required data. Memory locations may be read by scanning a load instruction into the CPU core which will reference the desired memory location, executing the load instruction, and then scanning out the result of the load. Other resources are accessed in a similar manner.

The initial entry by the CPU into the debug state (or mode) from normal, Waiting, Stopped, or Halted states (all indicated via the OnCE Status Register (OSR), OnCE Status Register) by assertion of one or more debug requests, begins a *debug session*. The **jd_debug_b** output signal indicates that a debug session is in progress, and the OSR will

indicate the CPU is in the debug state. Instructions may the be single-stepped by scanning new values into the CPUSCR, and performing a OnCE go+noexit command (See OnCE Command Register (OCMD)). The CPU will then temporarily exit the debug state (but not the debug session) to execute the instruction, and will then return to the debug state (again indicated via the OnCE Status Register (OSR)). The debug session remains in force until the final OnCE go+exit command is executed, at which time the CPU will return to the previous state it was in (unless a new debug request is pending). A scan into the CPUSCR is required prior to executing each go+exit or go+noexit OnCE command.

65.5.3 JTAG/OnCE Pins

The JTAG/OnCE pin interface is used to transfer OnCE instructions and data to the OnCE control block. Depending on the particular resource being accessed, the CPU may need to be placed in the Debug mode. For resources outside of the CPU block and contained in the OnCE block, the processor is not disturbed, and may continue execution. If a processor resource is required, an internal debug request (**dbg_dbgrq**) may be asserted to the CPU by the OnCE controller, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for further commands. Asserting **dbg_dbgrq** will cause the chip to temporarily exit the Waiting, Stopped or Halted power management states.

The following table details the primary JTAG/OnCE interface signals.

Table 65-22. JTAG/OnCE Primary Interface Signals

Signal name	Type	Description
j_trst_b	I	JTAG test reset
j_tclk	I	JTAG test clock
j_tms	I	JTAG test mode select
j_tdi[0:31]	I	JTAG test data or parallel input
j_tdo	O	Test data out to master controller or pad
j_tdo_en[1]	O	Enables TDO output buffer
j_parallel_IR_en	I	Enables parallel loading of IR during Shift_IR state
j_parallel_DR_en	I	Enables parallel loading of DR during Shift_DR state

1. j_tdo_en is asserted when the TAP controller is in the shift_DR or shift_IR state.

A full description of JTAG pins is provided in the JTAG Support Signals section of the Core (e200z4204n3) Core Complex Overview.

65.5.4 OnCE Internal Interface Signals

The following paragraphs describe the OnCE interface signals to other internal blocks associated with the OnCE controller.

65.5.4.1 CPU Debug Request (dbg_dbgrq)

The **dbg_dbgrq** signal is asserted by the OnCE control logic to request the CPU to enter the debug state. It may be asserted for a number of different conditions, and causes the CPU to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode, and wait for further commands.

65.5.4.2 CPU Debug Acknowledge (cpu_dbgack)

The **cpu_dbgack** signal is asserted by the CPU upon entering the debug state. This signal is used as part of the handshake mechanism between the OnCE control logic and the rest of the CPU. The CPU core may enter debug mode either through a software or hardware event.

65.5.4.3 CPU Address, Attributes

The CPU address and attribute information are used by the Nexus3 unit with information for real-time address trace information.

65.5.4.4 CPU Data

The CPU data buses are used to supply the Nexus3 debug unit with information for real-time data trace capability.

65.5.5 OnCE Interface Signals

The following paragraphs describe additional OnCE interface signals to other external blocks such as a Nexus controller and external blocks which may need information pertaining to debug operation.

65.5.5.1 OnCE Enable (jd_en_once)

The OnCE enable signal **jd_en_once** is used to enable the OnCE controller to allow certain instructions and operations to be executed. Assertion of this signal will enable the full OnCE command set, as well as operation of control signals and OnCE Control register functions. When this signal is disabled, only the Bypass, ID and Enable_OnCE commands are executed by the OnCE unit, and all other commands default to a "Bypass" command. The OnCE Status register (OSR) is not visible when OnCE operation is disabled. In addition, OnCE Control register (OCR) functions are disabled, as is the operation of the **jd_de_b** input. Secure systems may choose to leave the **jd_en_once** signal negated until a security check has been performed. Other systems should tie this signal asserted to enable full OnCE operation. The **j_en_once_regsel** output signal is provided to assist external logic performing security checks.

The **jd_en_once** input must only change state during the Test-Logic-Reset, Run-Test/ Idle, or Update_DR TAP states. A new value will take affect after one additional **j_tclk** cycle of synchronization. In addition, **jd_enable_once** input signal must not change state during a debug session, or undefined activity may occur.

65.5.5.2 OnCE Debug Request/Event (jd_de_b, jd_de_en)

If implemented at the SoC level, a system level bidirectional open drain debug event pin **DE_b** (not part of the e200z4204n3 interface) provides a fast means of entering the Debug Mode of operation from an external command controller (when input) as well as a fast means of acknowledging the entering of the Debug Mode of operation to an external command controller (when output). The assertion of this pin by a command controller causes the CPU core to finish the current instruction being executed, save the instruction pipeline information, enter Debug Mode, and wait for commands to be entered. If **DE_b** was used to enter the Debug Mode then **DE_b** must be negated after the OnCE controller responds with an acknowledge and before sending the first OnCE command. The assertion of this pin by the CPU Core acknowledges that it has entered the Debug Mode and is waiting for commands to be entered.

To support operation of this system pin, the OnCE logic supplies the **jd_de_en** output and samples the **jd_de_b** input when OnCE is enabled (**jd_en_once** asserted). Assertion of **jd_de_b** will cause the OnCE logic to place the CPU into Debug Mode. Once Debug Mode has been entered, the **jd_de_en** output will be asserted for three **j_tclk** periods to signal an acknowledge. **jd_de_en** can be used to enable the open-drain pulldown of the system level **DE_b** pin.

For systems which do not implement a system level bidirectional open drain debug event pin **DE_b**, the **jd_de_en** and **jd_de_b** signals may still be used to handshake debug entry.

65.5.5.3 OnCE Debug Output (jd_debug_b)

The OnCE Debug output **jd_debug_b** is used to indicate to on-chip resources that a debug session is in progress. Peripherals and other units may use this signal to modify normal operation for the duration of a debug session, which may involve the CPU executing a sequence of instructions solely for the purpose of visibility/system control which are not part of the normal instruction stream the CPU would have executed had it not been placed in debug mode. This signal is asserted the first time the CPU enters the debug state, and remains asserted until the CPU is released by a write to the OnCE Command Register with the GO and EX bits set, and a register specified as either "No Register Selected" or the CPUSCR. This signal will remain asserted even though the CPU may enter and exit the debug state for each instruction executed under control of the OnCE controller. See OnCE Command Register (OCMD) for more information on the function of the GO and EX bits. This signal is not normally used by the CPU.

65.5.5.4 CPU Clock On Input (jd_mclk_on)

The CPU Clock On input **jd_mclk_on** is used to indicate that the CPU's **m_clk** input is active. This input signal is expected to be driven by system logic external to the e200z4204n3 core, is synchronized to the **j_tclk** (scan clock) clock domain, and is presented as a status flag on the **j_tdo** output during the Shift_IR state. External firmware may use this signal to ensure proper scan sequences will occur to access debug resources in the **m_clk** clock domain.

65.5.5.5 Watchpoint Events (jd_watchpt[0:31])

The **jd_watchpt[0:31]** signals may be asserted by the OnCE control logic to signal that a watchpoint condition has occurred. Watchpoints do not directly cause the CPU to be affected. They are provided to allow external visibility only or for triggering purposes. Watchpoint events are conditioned by the settings in the DBCRxx registers, as well as by the DEVENT register, the DTC/DTSA/DTEA registers, and the Performance Monitor control register settings. Refer to Table 65-28 for details of the signal assignments. Note that assertion of most watchpoint outputs is conditioned on being in EDM or IDM. The Performance monitor, DEVENT, and DTC watchpoints are not conditioned however, and may assert regardless of the state of EDM or IDM. In addition, DAC1 or DAC3 watchpoints may be generated for stack limit violation occurrences when those resources are configured to perform stack limit checking, regardless of the state of EDM or IDM.

65.5.5.6 Update DR w/go+exit (j_ocmd_go_exit)

This signal indicates the TAP controller is in the Update_DR state and that the go and exit bits in the OnCE Command register are high, and RS indicates "no register selected". This signal will assert regardless of whether the CPU is currently in debug mode. It may be monitored by external logic to cause a synchronous exit from debug mode of other modules (but not other CPUs) in the system. A debugger can place all of the CPU tap controllers in parallel, scan in a go+exit command with "no register selected", and sequence through the Update_DR state to cause any CPU currently in debug mode to exit. CPUs which are not it debug mode will ignore the command.

65.5.6 OnCE Controller and serial interface

The OnCE Controller contains the OnCE command register, the OnCE decoder, and the status/control register. The following figure is a block diagram of the OnCE controller. In operation, the OnCE Command register acts as the IR for the TAP controller, and all other OnCE resources are treated as data registers (DR) by the TAP controller. The Command register is loaded by serially shifting in or parallel loading commands during the TAP controller Shift-IR state, and is loaded during the Update-IR state. The Command register selects a resource to be accessed as a data register (DR) during the TAP controller Capture-DR, Shift-DR and Update-DR states.

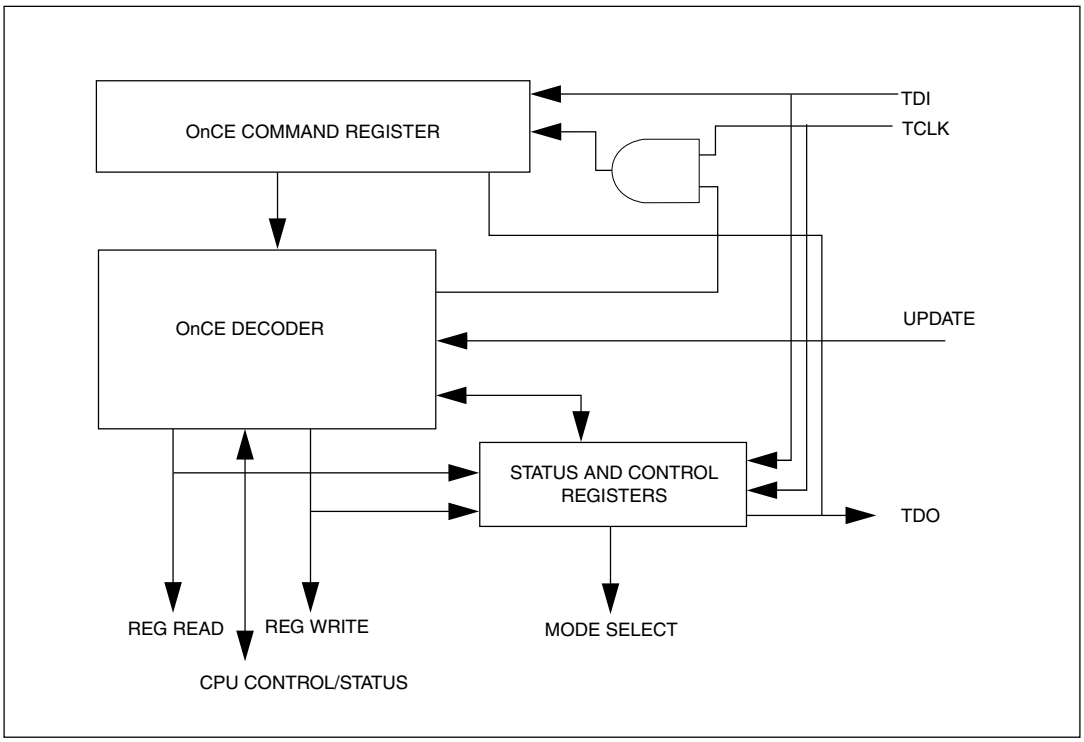


Figure 65-21. OnCE Controller and Serial Interface

65.5.6.1 OnCE Status Register

Status information regarding the state of the CPU is latched into the OnCE Status register when the OnCE controller state machine enters the Capture-IR state. When OnCE operation is enabled, this information is provided on the **j_tdo**[31] output in serial fashion when the Shift_IR state is entered following a Capture-IR. Information is shifted out least significant bit first.

Parallel output of the IR status (OSR) is available via the j_tdo[22:31] outputs during the first cycle of the Shift_IR state(s), with j_tdo[22] corresponding to the most-significant bit of the OSR (MCLK).

MCLK	ERR	0	RESET	HALT	STOP	DEBUG	WAIT	0	1
0	1	2	3	4	5	6	7	8	9

Figure 65-22. OnCE Status Register

The following table provides bit definitions for the Once Status Register.

Table 65-23. OnCE Status Register Bit Definitions

Bit(s)	Name	Description
0	MCLK	MCLK

Table continues on the next page...

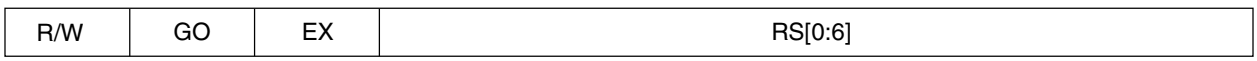
Table 65-23. OnCE Status Register Bit Definitions (continued)

Bit(s)	Name	Description
		m_clk Status Bit 0 - Inactive state 1 - Active state This status bit reflects the logic level on the **jd_mclk_on** input signal after capture by **j_tclk**.
1	ERR	ERROR This bit is used to indicate that an error condition occurred during attempted execution of the last single-stepped instruction (GO+NoExit with CPUSCR or No Register Selected in OCMD), and that the instruction may not have been properly executed. This could occur if an Interrupt (all classes including External, Critical, machine check, Storage, Alignment, Program, etc.) occurred while attempting to perform the instruction single step. In this case, the CPUSCR will contain information related to the first instruction of the Interrupt handler, and no portion of the handler will have been executed.
3	RESET	RESET Mode This bit reflects the inverted logic level on the CPU **p_reset_b** input after capture by **j_tclk**.
4	HALT	HALT Mode This bit reflects the logic level on the CPU **p_halted** output after capture by **j_tclk**.
5	STOP	STOP Mode This bit reflects the logic level on the CPU **p_stopped** output after capture by **j_tclk**.
6	DEBUG	Debug Mode This bit is asserted once the CPU is in debug mode. It is negated once the CPU exits debug mode (even during a debug session)
7	WAIT	Waiting Mode This bit reflects the logic level on the CPU **p_waiting** output after capture by **j_tclk**.
8	0	Reserved, set to 0 for 1149.1 compliance
9	1	Reserved, set to 1 for 1149.1 compliance

65.5.6.2 OnCE Command Register (OCMD)

The OnCE Command Register (OCMD) is a 10-bit shift register that receives its serial or parallel data from the TDI pins and serves as the instruction register (IR). It holds the 10-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in Figure 65-23. The OCMD is updated when the TAP controller enters the Update-IR state. It contains fields for controlling access to a resource, as well as controlling single-step operation and exit from OnCE mode.

Although the OCMD is updated during the Update-IR TAP controller state, the corresponding resource is accessed in the DR scan sequence of the TAP controller, and as such, the Update-DR state must be transitioned through in order for an access to occur. In addition, the Update-DR state must also be transitioned through in order for the single-step and/or exit functionality to be performed, even though the command appears to have no data resource requirement associated with it.



Reset - 10'b1000000010 on assertion of **j_trst_b** or **m_por** , or while in the Test_Logic_Reset state

Figure 65-23. OnCE Command Register

Table 65-24 provides bit definitions for the Once Command Register.

Table 65-24. OnCE Command Register Bit Definitions

Bit(s)	Name	Description
0	R/W	Read/Write Command Bit The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit. **Note:** The R/W bit generally ignored for read-only or write-only registers, although the PC FIFO pointer is only guaranteed to be update when R/W=1. In addition, it is ignored for all bypass operations. When performing writes, most registers are sampled in the Capture-DR state into a 32-bit shift register, and subsequently shifted out on **j_tdo[31]** during the first 32 clocks of Shift-DR, or on the j_tdo[0:31] outputs during the first clock of Shift-DR. 0 - Write the data associated with the command into the register specified by RS[0:6] 1 - Read the data contained in the register specified by RS[0:6]
1	GO	Go Go Command Bit If the GO bit is set, and the CPU is currently in debug mode, the CPU will execute the instruction which resides in the IR register in the CPUSCR. To execute the instruction, the processor leaves the debug mode, executes the instruction, and if the EX bit is cleared, returns to the debug mode immediately after executing the instruction. The processor goes on to normal operation if the EX bit is set, and no other debug request source is asserted. The GO command is executed only if the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the GO bit is ignored.The processor will leave the debug mode after the TAP controller Update-DR state is entered. On a GO+NoExit operation, returning to debug mode is treated as a debug event, thus exceptions such as machine checks and interrupts may take priority and prevent execution of the intended instruction. Debug firmware should mask these exceptions as appropriate. The OSR_{ERR} bit indicates such an occurrence. If the CPU is not currently in debug mode, then the GO command will be ignored. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking. 0 - Inactive (no action taken) 1 - Execute instruction in IR

Table continues on the next page...

Table 65-24. OnCE Command Register Bit Definitions (continued)

Bit(s)	Name	Description
2	EX	Exit Command Bit 0 - Remain in debug mode 1 - Leave debug mode If the EX bit is set, the processor will leave the debug mode and resume normal operation until another debug request is generated. The Exit command is executed only if the Go command is issued, and the operation is a read/write to CPUSCR or a read/write to "No Register Selected". Otherwise the EX bit is ignored. The processor will leave the debug mode after the TAP controller Update-DR state is entered. Note that if the DR bit in the OnCE control register is set or remains set, or if a bit in EDBSR0 is set and $EDBCR0_{EDM}$=1 (external debug mode is enabled), or if another debug request source is asserted, then the processor may return to the debug mode *without* execution of an instruction, even though the EX bit was set. Note that asynchronous interrupts are blocked on a GO+Exit operation until the first instruction to be executed begins execution. See Exiting Debug Mode and Interrupt Blocking.
3:9	RS	Register Select The Register Select bits define which register is source (destination) for the read (write) operation. Table 65-25 indicates the OnCE register addresses. Attempted writes to read-only registers are ignored.

Table 65-25 indicates the OnCE register addresses.

Table 65-25. OnCE Register Addressing

RS[0:6]	Register Selected
000 0000	Reserved
000 0001	Reserved
000 0010	JTAG ID (read-only)
000 0011–000 1111	Reserved
001 0000	CPU Scan Register (CPUSCR)
001 0001	No Register Selected (Bypass)
001 0010	OnCE Control Register (OCR)
001 0011	Reserved
001 0100–001 0111	Reserved
001 1000	Data Address Compare 3 (DAC3)
001 1001	Data Address Compare 4 (DAC4)
001 1010–001 1111	Reserved
010 0000	Instruction Address Compare 1 (IAC1)
010 0001	Instruction Address Compare 2 (IAC2)
010 0010	Instruction Address Compare 3 (IAC3)
010 0011	Instruction Address Compare 4 (IAC4)
010 0100	Data Address Compare 1 (DAC1)
010 0101	Data Address Compare 2 (DAC2)

Table continues on the next page...

Table 65-25. OnCE Register Addressing (continued)

RS[0:6]	Register Selected
010 0110	Data Value Compare 1 (DVC1) - *all 64 bits of the DVC register are accessed
010 0111	Data Value Compare 2 (DVC2) - *all 64 bits of the DVC register are accessed
010 1000	Instruction Address Compare 5 (IAC5)
010 1001	Instruction Address Compare 6 (IAC6)
010 1010	Instruction Address Compare 7 (IAC7)
010 1011	Instruction Address Compare 8 (IAC8)
010 1100	Debug Data Effective Address (DDEAR)
010 1101	External Debug Data Effective Address (EDDEAR)
010 1110	External Debug Control Register 0 (EDBCR0)
010 1111	External Debug Status Register 0 (EDBSR0)
011 0000	Debug Status Register (DBSR)
011 0001	Debug Control Register 0 (DBCR0)
011 0010	Debug Control Register 1 (DBCR1)
011 0011	Debug Control Register 2 (DBCR2)
011 0100	Reserved (do not access)
011 0101	Debug Control Register 4 (DBCR4)
011 0110	Debug Control Register 5 (DBCR5)
011 0111	Debug Control Register 6 (DBCR6)
011 1000	Debug Control Register 7 (DBCR7)
011 1001	Debug Control Register 8 (DBCR8)
011 1010	Reserved (do not access)
011 1011	Reserved (do not access)
011 1100	External Debug Status Register MASK 0 (EDBSRMSK0)
011 1101	Debug Data Acquisition Message Register (DDAM)
011 1110	Debug Event Control (DEVENT)
011 1111	External Debug Resource Allocation Control 0 (EDBRAC0)
100 0000	Debug L1 Cache Control/Status 0 (DBL1CCSR0)
100 0001–110 1100	Reserved (do not access)
110 1101	MPU0 Control/Status 0 (MPU0CSR0)
110 1110	Performance Monitor Register Access
110 1111	Reserved for Shared Nexus Control Register Select
111 0000–111 1001	General Purpose register selects [0:9]
111 1010	Cache Debug Access Control Register (CDACNTL)
111 1011	Cache Debug Access Data Register (CDADATA)
111 1100	Nexus3-Access (<<<put in cross reference to Nexus 3 Module>>>)
111 1101	LSRL Select (see Test Specification)
111 1110	Enable_OnCE[1]
111 1111	Bypass

1. Causes assertion of the j_en_once_regsel output.

The Once Decoder receives as input the 10-bit command from the OCMD, and status signals from the processor, and generates all the strobes required for reading and writing the selected OnCE registers.

Single stepping of instructions is performed by placing the CPU in debug mode, scanning in appropriate information into the CPUSCR, and setting the Go bit (with the EX bit cleared) with the RS field indicating either the CPUSCR or No Register Selected. After executing a single instruction, the CPU will re-enter debug mode and await further commands. During single-stepping, exception conditions may occur if not properly masked by debug firmware (interrupts, machine checks, bus error conditions, etc.) and may prevent the desired instruction from being successfully executed. The OSR_{ERR} bit is set to indicate this condition. In these cases, values in the CPUSCR will correspond to the first instruction of the exception handler.

Additionally, the $EDBCR0_{EDM}$ bit is forced to '1' internally while single-stepping to prevent Debug events from generating Debug interrupts. Also, during a debug session, the DBSR register is frozen from updates due to debug events other than execution of a DNI-type instruction, regardless of EDBCR0EDM. DBSR may still be modified during a debug session via a single-stepped **mtspr** instruction, or via OnCE access.

If the CPU is not currently in debug mode, go+exit and go+noexit commands are ignored, although for a go+exit command with "No Register Selected", the j_ocmd_go_exit output will be asserted.

65.5.6.3 OnCE Control Register (OCR)

The OnCE Control Register is a 32-bit register used to force the e200z4204n3 core into debug mode and to enable / disable sections of the OnCE control logic. The control bits are read/write. These bits are only effective while OnCE is enabled (**jd_en_once** asserted). The OCR is shown in the following figure.

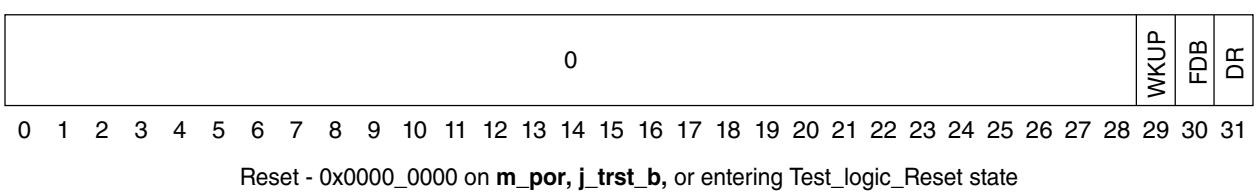


Figure 65-24. OnCE Control Register

The following table provides bit definitions for the OnCE Control Register.

Table 65-26. OnCE Control Register Bit Definitions

Bit(s)	Name	Description
0:28	—	Reserved
29	WKUP	Wakeup Request Bit (WKUP) This control bit may be used to force the **p_wakeup** output signal to be asserted. This control function may be used by debug firmware to request that the chip-level clock controller restore the **m_clk** input to normal operation regardless of whether the CPU is in a low power state to ensure that debug resources may be properly accessed by external hardware through scan sequences.
30	FDB	Force Breakpoint Debug Mode Bit (FDB) This control bit is used to determine whether the processor is operating in breakpoint debug enable mode or not. The processor may be placed in breakpoint debug enable mode by setting this bit. In breakpoint debug enable mode, execution of the '**bkpt**' pseudo- instruction will cause the processor to enter debug mode, as if the **jd_de_b** input had been asserted. This bit is qualified with $EDBCR0_{EDM}$, which must be set for FDB to take effect. Note that this bit has no effect on **e_dnh** or **se_dnh** instruction operation.
31	DR	CPU Debug Request Control Bit This control bit is used to unconditionally request the CPU to enter the Debug Mode. The CPU will indicate that Debug Mode has been entered via the data scanned out in the shift-IR state. 0 - No Debug Mode request 1 - Unconditional Debug Mode request When the DR bit is set the processor will enter Debug mode at the next instruction boundary.

65.5.7 Access to Debug Resources

Resources contained in the OnCE Module which do not require the processor core to be halted for access may be accessed while the e200z4204n3 core is running, and will not interfere with processor execution. Accesses to other resources such as the CPUSCR require the e200z4204n3 core to be placed in debug mode to avoid synchronization hazards. Debug firmware may ensure that it is safe to access these resources by determining the state of the e200z4204n3 core prior to access. Note that a scan operation to update the CPUSCR is required prior to exiting debug mode if debug mode has been entered.

Some cases of write accesses other than accesses to the OnCE Command and Control registers, or the EDM bit of DBCR0 require **m_clk** to be running for proper operation. The OnCE control register provides a means of signaling this need to a system level clock control module via the OCR_{WKUP} control bit.

In addition, since the CPU may cause multiple bits of certain registers to change state, reads of certain registers while the CPU is running (DBSR, etc.) may not have consistent bit settings unless read twice with the same value indicated. In order to guarantee that the contents are consistent, the CPU should be placed into debug mode, or multiple reads should be performed until consistent values have been obtained on consecutive reads.

The following table provides a list of access requirements for OnCE registers.

Table 65-27. OnCE Register Access Requirements

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 $_{EDM}$ = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
Enable_OnCE	N	N	N	N	—	
Bypass	N	N	N	N	N	
CPUSCR	Y	Y	Y	Y	Y	
DAC1–4	Y	Y	Y	N	*[1]	
DBCR0	Y	Y	Y	N	*[1]	*EDBCR0$_{EDM}$ access only requires **jd_en_once** asserted
DBCR1–8	Y	Y	Y	N	*[1]	
DEVENT	Y	Y	Y	N	*[1]	
EDBRAC0 (DBERC0)	Y	N	Y	N	*[1]	
DBSR	Y	Y	Y	N[2]	*[1]	
EDBCR0	Y	N	N	N	N	
EDBSR0	Y	N	Y	N	N	
EDBSRMSK0	Y	N	N	N	N	
IAC1–8	Y	Y	Y	N	*[1]	
JTAG ID	N	N	—	N	—	Read-only
OCR	Y	N	N	N	N	
OSR	Y	N	—	N	—	Read-only, accessed by scanning out IR while **jd_en_once** is asserted
Cache Debug Access Control (CDACNTL)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Cache Debug Access Data (CDADATA)	Y	N	Y	N	N	CPU gives lowest priority to a Cache access request when it is not debug halted
Nexus3-Access	Y	N	N	N	N	
PMR-Access	Y	N	N	N	N	
PMR Registers	Y	Y	Y	N	N	
External GPRs	Y	N	N	N	N	

Table continues on the next page...

Table 65-27. OnCE Register Access Requirements (continued)

Register name	Access requirements					Notes
	Requires jd_en_once to be asserted	Requires EDBCR0 EDM = 1 for write access	Requires m_clk active for Write Access	Requires CPU to be halted for Read Access	Requires CPU to be halted for Write Access	
LSRL Select	Y	N	?	?	?	System Test logic implementation determines LSRL functionality

1. Writes to these registers while the CPU is running may have unpredictable results due to the pipelined nature of operation, and the fact that updates are not synchronized to a particular clock, instruction, or bus cycle boundary, therefore it is strongly recommended to ensure the processor is first placed into debug mode before updates to these registers are performed.
2. Reads of these registers while the CPU is running may not give data that is self-consistent due to synchronization across clock domains.

65.5.8 Methods of Entering Debug Mode

The OnCE Status Register indicates that the CPU has entered the debug mode via the DEBUG status bit. The following sections describe how Debug Mode is entered assuming the OnCE circuitry has been enabled. OnCE operation is enabled by the assertion of the **jd_en_once** input See OnCE Enable (jd_en_once).

65.5.8.1 External Debug Request During RESET

Holding the **jd_de_b** signal asserted during the assertion of **p_reset_b,** and continuing to hold it asserted following the negation of **p_reset_b** causes the e200z4204n3 core to enter Debug Mode. After receiving an acknowledge via the OnCE Status Register DEBUG bit, the external command controller should negate the **jd_de_b** signal before sending the first command. Note that in this case the e200z4204n3 core does not execute an instruction before entering Debug Mode, although the first instruction to be executed will be fetched prior to entering Debug Mode in order to properly initialize the CPUSCR.

65.5.8.2 Debug Request During RESET

Asserting a debug request by setting the DR bit in the OCR during the assertion of **p_reset_b** and then negating **p_reset_b** causes the chip to enter debug mode. In this case the CPU will fetch the first instruction of the reset exception handler in order to properly initialize the CPUSCR, but does not execute an instruction before entering debug mode.

65.5.8.3 Debug Request During Normal Activity

Asserting a debug request by setting the DR bit in the OCR during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. Note that in this case the chip completes the execution of the current instruction and stops after the newly fetched instruction enters the CPU instruction register. This process is the same for any newly fetched instruction including instructions fetched by the interrupt processing, or those that will be aborted by the interrupt processing.

65.5.8.4 Debug Request During Waiting, Halted or Stopped State

Asserting a debug request by setting the DR bit in the OCR when the chip is in the Waiting state (**p_waiting** asserted), Halted state (**p_halted** asserted) or Stopped state (**p_stopped** asserted) causes the CPU to exit the state and enter the debug mode once the CPU clock **m_clk** has been restored. Note that in this case, the CPU will negate the **p_waiting, p_halted** and **p_stopped** outputs. Once the debug session has ended, the CPU will return to the state it was in prior to entering debug mode.

To signal the chip-level clock generator to re-enable **m_clk**, the **p_wakeup** output will be asserted whenever the debug block is asserting a debug request to the CPU due to OCR_{DR} being set, or **jd_de_b** assertion, and will remain set from then until the debug session ends (**jd_debug_b** goes from asserted to negated). In addition, the status of the **jd_mclk_on** input (after synchronization to the **j_tclk** clock domain) may be sampled along with other status bits from the **j_tdo** outputs during the Shift_IR TAP controller state. This status may be used if necessary by external debug firmware to ensure proper scan sequences occur to registers in the **m_clk** clock domain.

65.5.8.5 Software Request During Normal Activity

Upon executing a '**bkpt**' pseudo-instruction (for e200z4204n3, defined to be an all 0's instruction opcode) when the OCR register's (FDB) bit is set (debug mode enable control bit is true), and $EDBCR0_{EDM}$=1, the CPU enters the debug mode after the instruction following the '**bkpt**' pseudo-instruction has entered the instruction register.

65.5.8.6 Debug Notify Halt Instructions

The **e_dnh** and **se_dnh** instructions allow software to transition the core from a running state to a debug halted state if enabled by $EDBCR0_{DNH_EN}$, and provide the external debugger with bits reserved in the instruction itself to pass additional information. Entry into debug mode is ***not*** conditioned on $EDBCR0_{EDM}$, allowing for debug of software debug handlers as well as other software. For e200z4204n3, when the CPU enters a debug halted state due to a **e_dnh** or **se_dnh** instruction, the instruction will be stored in the CPUSCR[IR] portion, and the CPUSCR[PC] value will point to the instruction. The external debugger should update the CPUSCR prior to exiting the debug halted state to point past the **e_dnh** or **se_dnh** instruction.

65.5.9 CPU Status and Control Scan Chain Register (CPUSCR)

A number of on-chip registers store the CPU pipeline status and are configured in a single scan chain for access by the OnCE controller. The CPUSCR register contains these processor resources, which are used to restore the pipeline and resume normal chip activity upon return from the debug mode, as well as a mechanism for the emulator software to access processor and memory contents. The following figure shows the block diagram of the pipeline information registers contained in the CPUSCR. Once debug mode has been entered, it is required to scan in and update this register prior to exiting debug mode.

NOTE

Note that due to the length of this register, parallel JTAG access capability is not provided.

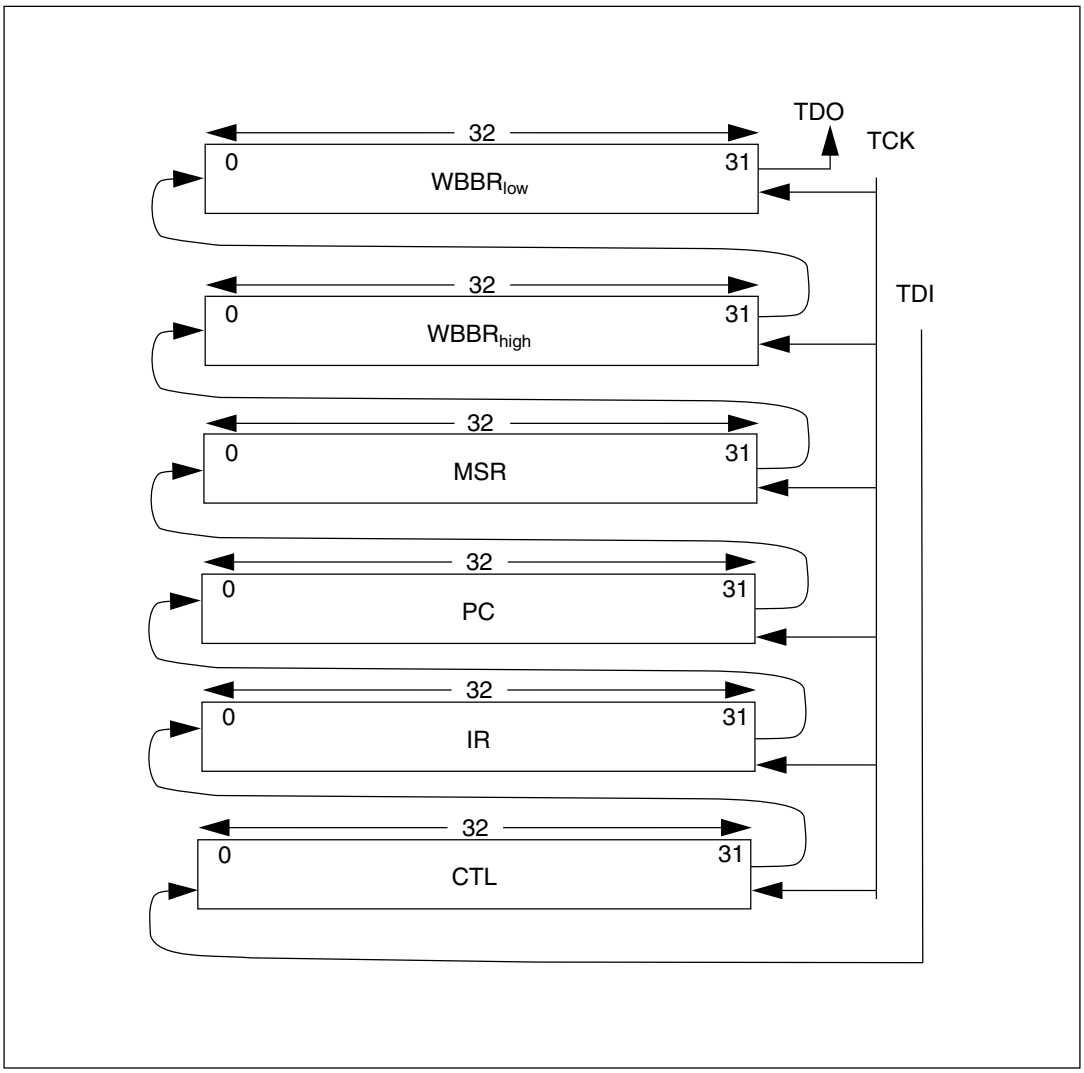


Figure 65-25. CPU Scan Chain Register (CPUSCR)

65.5.9.1 Instruction Register (IR)

The Instruction Register (IR) provides a mechanism for controlling the debug session by serving as a means for forcing in selected instructions, and then causing them to be executed in a controlled manner by the debug control block. The opcode of the next instruction to be executed when entering debug mode is contained in this register when the scan-out of this chain begins. This value should be saved for later restoration if continuation of the normal instruction stream is desired.

On scan-in, in preparation for exiting debug mode, this register is filled with an instruction opcode selected by debug control software. By selecting appropriate instructions and controlling the execution of those instructions, the results of execution may be used to examine or change memory locations and processor registers. The debug

control module external to the processor core will control execution by providing a single-step capability. Once the debug session is complete and normal processing is to be resumed, this register may be loaded with the value originally scanned out.

65.5.9.2 Control State Register (CTL)

The Control State Register (CTL) is a 32-bit register that stores the value of certain internal CPU state variables before the debug mode is entered. This register is affected by the operations performed during the debug session and should normally be restored by the external command controller when returning to normal mode. In addition to saved internal state variables, two of the bits are used by emulation firmware to control the debug process. In certain circumstances, emulation firmware must modify the content of this register as well as the PC and IR values in the CPUSCR before exiting debug mode. These cases are described below.

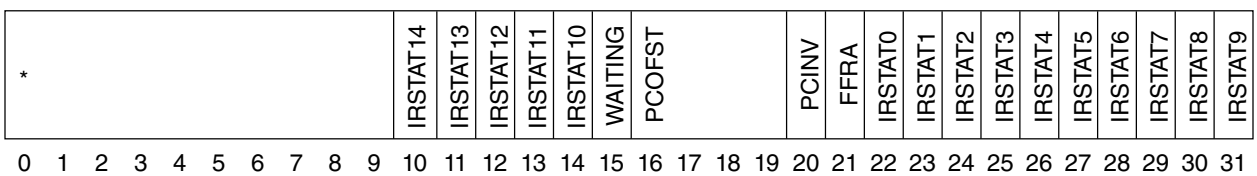


Figure 65-26. Control State Register (CTL)

WAITING — WAITING State Status

This bit indicates whether the CPU was in the waiting state prior to entering debug mode. If set, the CPU was in the waiting state. Upon exiting a debug session, the value of this bit in the restored CPUSCR will determine whether the CPU re-enters the waiting state on a go+exit.

0- CPU was not in the waiting state when debug mode was entered

1- CPU was in the waiting state when debug mode was entered

PCOFST — PC Offset Field

This field indicates whether the value in the PC portion of the CPUSCR must be adjusted prior to exiting debug mode. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored into the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction instead of the original IR value, other wise the original value of IR should be restored. (But see PCINV which overrides this field)

0000 - No correction required.

0001 - Subtract 0x04 from PC.

0010 - Subtract 0x08 from PC.

0011 - Subtract 0x0C from PC.

0100 - Subtract 0x10 from PC.

0101 - Subtract 0x14 from PC.

all other encodings are reserved

* — Internal State Bits

Table continues on the next page...

These control bits represent internal processor state and should be restored to their original value after a debug session is completed, i.e when a OnCE command is issued with the GO and EX bits set and not ignored. When performing instruction execution during a debug session. See OnCE Debug Output (jd_debug_b), which is not part of the normal program execution flow, these bits should be set to a 0.

PCINV — PC and IR Invalid Status Bit

This status bit indicates that the values in the IR and PC portions of the CPUSCR are invalid. Exiting debug mode with the saved values in the PC and IR will have unpredictable results. Debug firmware should initialize the PC and IR values in the CPUSCR with desired values prior to exiting debug mode if this bit was set when debug mode was initially entered.

0= No error condition exists.

1= Error condition exists. PC and IR are corrupted.

FFRA— Feed Forward RA Operand Bit

This control bit causes the content of the $WBBR_{lo}$ to be used as the RA operand value (RS for logical, mtspr, mtdcr, cntlzw, and shift operations, RX for VLE se_ instructions, RT for e_{logical_op}2i type instructions, RB for evaddiw, evsubifw, and the value to use as the PC for calculating the LR update value for branch with link type instructions) of the first instruction to be executed following an update of the CPUSCR.

This allows the debug firmware to update processor registers — initialize the $WBBR_{lo}$ with the desired value, set the FFRA bit, and execute a ori Rx,Rx,0 instruction to the desired register.

Note: not all instructions support using the FFRA control. FFRA is mainly intended for use with the ori instruction to allow the debugger to write to a GPR. Support for other instructions is implementation-dependent.

0= No action.

1= Content of $WBBR_{lo}$ used as operand value

IRStat0 — IR Status Bit 0

This control bit indicates a TEA status for the IR.

0= No TEA occurred on the fetch of this instruction.

1= TEA occurred on the fetch of this instruction.

IRStat1 — IR Status Bit 1

This control bit is reserved.

IRStat2 — IR Status Bit 2

This control bit indicates an Instruction Address Compare 1 event status for the IR.

0= No Instruction Address Compare event 1 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 1 occurred on the fetch of this instruction.

IRStat3 — IR Status Bit 3

This control bit indicates an Instruction Address Compare 2 event status for the IR.

0= No Instruction Address Compare 2 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 2 event occurred on the fetch of this instruction.

IRStat4 — IR Status Bit 4

This control bit indicates an Instruction Address Compare 3 event status for the IR.

0= No Instruction Address Compare event 3 occurred on the fetch of this instruction.

1= An Instruction Address Compare event 3 occurred on the fetch of this instruction.

IRStat5 — IR Status Bit 5

This control bit indicates an Instruction Address Compare 4 event status for the IR.

Table continues on the next page...

0= No Instruction Address Compare 4 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 4 event occurred on the fetch of this instruction.

IRStat6 — IR Status Bit 6

This control bit indicates a Parity Error status for the IR.

0= No Parity Error occurred on the fetch of this instruction.

1= Parity Error occurred on the fetch of this instruction from the I-Cache.

IRStat7 — IR Status Bit 7

This control bit indicates a Precise External Termination Error status for the IR.

0= No Precise External Termination Error occurred on the fetch of this instruction.

1= A Precise External Termination Error occurred on the fetch of this instruction.

IRStat8 — IR Status Bit 8

This control bit indicates the PowerISA VLE status for the IR.

0= IR contains a BookE instruction. (unused encoding)

1= IR contains a PowerISA VLE instruction, aligned in the Most Significant Portion of IR if 16-bit.

- this bit should not be modified by the debugger, otherwise the resulting operation is boundedly undefined. It should always indicate VLE (=1).

IRStat9 — IR Status Bit 9

This control bit is reserved.

IRStat10 — IR Status Bit 10

This control bit indicates an Instruction Address Compare 5 event status for the IR.

0= No Instruction Address Compare 5 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 5 event occurred on the fetch of this instruction.

IRStat11 — IR Status Bit 11

This control bit indicates an Instruction Address Compare 6 event status for the IR.

0= No Instruction Address Compare 6 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 6 event occurred on the fetch of this instruction.

IRStat12 — IR Status Bit 12

This control bit indicates an Instruction Address Compare 7 event status for the IR.

0= No Instruction Address Compare 7 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 7 event occurred on the fetch of this instruction.

IRStat13 — IR Status Bit 13

This control bit indicates an Instruction Address Compare 8 event status for the IR.

0= No Instruction Address Compare 8 event occurred on the fetch of this instruction.

1= An Instruction Address Compare 8 event occurred on the fetch of this instruction.

IRStat14 — IR Status Bit 14

This control bit is reserved.

Emulation firmware should modify the content of the CTL, PC, and IR values in the CPUSCR during execution of debug related instructions as well as just prior to exiting debug with a go+exit command. During the debug session, the CTL register should be

written with the FFRA bit set as appropriate, and all other bit set to '0', and the IR set to the value of the desired instruction to be executed. IRStat8 will be used to determine the type of instruction present in the IR.

Just prior to exiting debug mode with a go+exit, the PCINV status bit which was originally present when debug mode was first entered should be tested, and if set, the PC and IR initialized for performing whatever recovery sequence is appropriate for a faulted exception vector fetch. If the PCINV bit is cleared, then the PCOFST bits should be examined to determine whether the PC value must be adjusted. Due to the pipelined nature of the CPU, the PC value must be backed-up by emulation software in certain circumstances. The PCOFST field specifies the value to be subtracted from the original value of the PC. This adjusted PC value should be restored in to the PC portion of the CPUSCR just prior to exiting debug mode with a go+exit. In the event the PCOFST is non-zero, the IR should be loaded with a nop instruction (such as **ori r0,r0,0**) instead of the original IR value, otherwise the original value of IR should be restored. Note that when a correction is made to the PC value, it will generally point to the last completed instruction, although that instruction will not be re-executed. The nop instruction is executed instead, and instruction fetch and execution will resume at location PC+4. IRStat8 will be used to determine the type of instruction present in the IR, thus should be cleared in this case. Note that debug events which may occur on the nop (ICMP) will be generated if enabled.

For the CTL register, the internal state bits should be restored to their original value. The IRStatus bits should be set to '0's if the PC was adjusted. If no PC adjustment was performed, emulation firmware should determine whether other IRStat flags should be set to '0' to avoid re-entry into debug mode for an instruction breakpoint request. Upon exiting debug mode with go+exit, if one of these bits is set, debug mode will be re-entered prior to any further instruction execution.

65.5.9.3 Program Counter Register (PC)

The PC is a 32-bit register that stores the value of the program counter which was present when the chip entered the debug mode. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the CPU returns to normal mode. PC normally points to the instruction contained in the IR portion of CPUSCR. If debug firmware wishes to redirect program flow to an arbitrary location, the PC and IR should be initialized to correspond to the first instruction to be executed upon resumption of normal processing. Alternatively, the IR may be set to a nop and the PC set to point to the location prior to the location at which it is desired to redirect flow to. On exiting debug mode, the nop will be executed, and instruction fetch and execution will resume at PC+4.

65.5.9.4 Write-Back Bus Register ($WBBR_{low}$, $WBBR_{high}$)

WBBR is used as a means of passing operand information between the CPU and the external command controller. Whenever the external command controller needs to read the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to WBBR. $WBBR_{low}$ holds the 32-bit result of most instructions including load data returned for a load or load with update instruction. $WBBR_{high}$ holds the updated effective address calculated by a load with update instruction. It is undefined for other instructions.

As an example, to read the 32 bits of processor register **r1**, an **ori r1,r1,0** instruction is executed, and the result value of the instruction will be latched into $WBBR_{low}$. The contents of $WBBR_{low}$ can then be delivered serially to the external command controller. To update a processor resource, this register is initialized with a data value to be written, and an **ori** instruction is executed which uses this value as a substitute data value. The Control State register FFRA bit forces the value of the $WBBR_{low}$ to be substituted for the normal RS source value of the **ori** instruction, thus allowing updates to processor registers to be performed. (Refer to Control State Register (CTL) for more detail on the CTL_{FFRA} bit.)

$WBBR_{low}$ and $WBBR_{high}$ are generally undefined on instructions which do not writeback a result, and due to control issues are not defined on **lmw** or branch instructions as well.

65.5.9.5 Machine State Register (MSR)

The MSR is a 32-bit register used to read/write the Machine State Register. Whenever the external command controller needs to save or modify the contents of the Machine State Register, this register is used.This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode.

65.5.9.6 Exiting Debug Mode and Interrupt Blocking

When exiting debug mode with a Go+Exit, "asynchronous" interrupts are blocked until the first instruction to be executed begins execution. This includes External and Critical input, NMI, and machine check interrupts. Asynchronous debug interrupts are not blocked however, and the CPU will re-enter debug mode without executing an instruction

following Go+Exit, although it may fetch an instruction and discard it. Exceptions due to an illegal instruction or error flags set within the CPUSCR CTL register are not blocked, since they apply to the instruction in the CPUSCR IR.

65.5.10 Reserved Registers (Reserved)

The Reserved Registers are used to control various test control logic. These registers are not intended for customer use. To preclude device and/or system damage, these registers should not be accessed.

65.6 Watchpoint Support

e200z4204n3 supports the generation and signalling of watchpoints when operating in internal debug mode ($DBCR0_{IDM}$=1) or in external debug mode ($EDBCR0_{EDM}$=1). Watchpoints are indicated with a dedicated set of interface signals. The **jd_watchpt[0:31]** output signals are used to indicate that a watchpoint has occurred. Certain watchpoints however (DEVENT-based, DTC-based, and Performance Monitor watchpoints) are not qualified with $EDBCR0_{EDM}$ or $DBCR0_{IDM}$.

Each debug address compare function (IAC1–8, DAC1–4), as well as other event types are capable of triggering a watchpoint output. The DBCRx control fields are used to configure watchpoints, regardless of whether events are enabled in DBCR0. Watchpoints may occur whenever an associated event would have been posted in the Debug Status Register if enabled. No explicit enable bits are provided for watchpoints; they are always enabled by definition, although the data address compare watchpoints may be controlled for read and write accesses via configuration fields in DBCR4, 7, and 9. During a debug session, debug events (other than PMI, DEVT1 and DEVT2) with a corresponding DBSR bit are blocked from asserting a watchpoint. The Performance Monitor, DEVNT-based and DTC-based watchpoints are not blocked during a debug session. If not desired, for address-based events the base address values for these events may be programmed to an unused system address. MSR_{DE} has no effect on watchpoint generation.

External logic may monitor the assertion of these signals for debugging or triggering purposes. Watchpoints are signaled in the clock cycle following the occurrence of the actual event. The Nexus3 module also monitors assertion of a portion of these output signals (**jd_watchpt[0:31]**) for various development control purposes. (See the "Watchpoint Trace Messaging" section in the Core (e200z4204n3) Nexus 3 Module chapter.)

Note that these signals are **m_clk** domain signals, not **j_tclk** domain signals.

The following table shows the **jd_watchpt[0:31]** output signal assignments to various debug events.

Table 65-28. Watchpoint Output Signal Assignments

Signal Name	Type	Description
jd_watchpt[0]	IAC1	Instruction Address Compare 1 watchpoint Asserted whenever an IAC1 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[1]	IAC2	Instruction Address Compare 2 watchpoint Asserted whenever an IAC2 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[2]	IAC3	Instruction Address Compare 3 watchpoint Asserted whenever an IAC3 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[3]	IAC4	Instruction Address Compare 4 watchpoint Asserted whenever an IAC4 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[4]	DAC1	Data Address Compare 1 watchpoint Asserted whenever a DAC1 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC1CFG}$ settings
jd_watchpt[5]	DAC2	Data Address Compare 2 watchpoint Asserted whenever a DAC2 compare occurs regardless of being enabled to set DBSR status, based on $DBCR4_{DAC2CFG}$ settings
jd_watchpt[6]	DAC3	Data Address Compare 3 watchpoint Asserted whenever a DAC3 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC3CFG}$ settings
jd_watchpt[7]	DAC4	Data Address Compare 4 watchpoint Asserted whenever a DAC4 compare occurs regardless of being enabled to set DBSR status, based on $DBCR7_{DAC4CFG}$ settings
jd_watchpt[8]	IAC5	Instruction Address Compare 5 watchpoint Asserted whenever an IAC5 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[9]	IAC6	Instruction Address Compare 6 watchpoint Asserted whenever an IAC6 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[10]	DEVT1	Debug Event Input 1 watchpoint Asserted whenever a DEVT1 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[11]	DEVT2	Debug Event Input 2 watchpoint Asserted whenever a DEVT2 debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[12]	DEVNT0	Debug Event Output 0 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[12]
jd_watchpt[13]	DEVNT1	Debug Event Output 1 watchpoint

Table continues on the next page...

Table 65-28. Watchpoint Output Signal Assignments (continued)

Signal Name	Type	Description
		Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[13]
jd_watchpt[14]	IAC7	Instruction Address Compare 7 watchpoint Asserted whenever an IAC7 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[15]	IAC8	Instruction Address Compare 8 watchpoint Asserted whenever an IAC8 compare occurs regardless of being enabled to set DBSR status
jd_watchpt[16]	IRPT	Interrupt watchpoint Asserted whenever an IRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[17]	RET	Return watchpoint Asserted whenever a RET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[18]	CIRPT	Critical Interrupt watchpoint Asserted whenever a CIRPT debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[19]	CRET	Critical Return watchpoint Asserted whenever a CRET debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[20]	DEVNT2	Debug Event Output 2 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[20]
jd_watchpt[21]	DEVNT3	Debug Event Output 3 watchpoint Asserted whenever a '1' is written to the bit of the DEVNT field of the DEVENT debug register corresponding to jd_watchpt[21]
jd_watchpt[22]	PMEVENT	Performance Monitor Event input watchpoint Asserted whenever **p_pm_event** transitions from a '0' to a '1' while **m_clk** is running
jd_watchpt[23]	PMC0	Performance Monitor Counter 0 watchpoint Asserted whenever PMC0 triggers an event based on $PMLCa0_{PMP}$
jd_watchpt[24]	PMC1	Performance Monitor Counter 1 watchpoint Asserted whenever PMC1 triggers an event based on $PMLCa1_{PMP}$
jd_watchpt[25]	PMC2	Performance Monitor Counter 2 watchpoint Asserted whenever PMC2 triggers an event based on $PMLCa2_{PMP}$
jd_watchpt[26]	PMC3	Performance Monitor Counter 3 watchpoint Asserted whenever PMC3 triggers an event based on $PMLCa3_{PMP}$
jd_watchpt[27]	—	Reserved
jd_watchpt[28]	TRAP	TRAP watchpoint Asserted whenever an TRAP debug event occurs regardless of being enabled to set DBSR status
jd_watchpt[29]	DTC1	Data Trace Control Range 1 watchpoint

Table continues on the next page...

Table 65-28. Watchpoint Output Signal Assignments (continued)

Signal Name	Type	Description
		Asserted whenever an access meets the conditions for DTC Range 1
jd_watchpt[30]	DTC2	Data Trace Control Range 2 watchpoint Asserted whenever an access meets the conditions for DTC Range 2
jd_watchpt[31]	DTC3	Data Trace Control Range 3 watchpoint Asserted whenever an access meets the conditions for DTC Range 3

65.7 Cache Array Access During Debug

The cache arrays may be read and written during debug mode via the CDACNTL and CDADATA debug registers.

65.8 Basic Steps for Enabling, Using, and Exiting External Debug Mode

The following steps show one possible scenario for a debugger wishing to use the external debug facilities. *This simplified flow is intended to illustrate basic operations, but does not cover all potential methods in depth.*

Enabling External Debug Mode and initializing Debug registers

- The debugger should ensure that the **jd_en_once** control signal is asserted in order to enable OnCE operation
- Select the OCR and write a value to it in which OCR_{DR}, OCR_{WKUP}, are set to '1'. The tap controller must step through the proper states as outlined earlier. This step will place the CPU in a debug state in which it is halted and awaiting single-step commands or a release to normal mode
- Scan out the value of the OSR to determine that the CPU clock is running and the CPU has entered the Debug state. This can be done in conjunction with a Read of the CPUSCR. The OSR is shifted out during the Shift_IR state. The CPUSCR will be shifted out during the Shift_DR state. The debugger should save the scanned-out value of CPUSCR for later restoration.
- Select the DBCR0 register and update it with the $EDBCR0_{EDM}$ bit set

- Clear the DBSR status bits
- Write appropriate values to the DBCR0–8, IAC, and DAC registers. Note that the initial write to DBCR0 will only affect the EDM bit, so the remaining portion of the register must now be initialized, keeping the EDM bit set

At this point the system is ready to commence debug operations. Depending on the desired operation, different steps must occur.

- Optionally, ensure that the values entered into the MSR portion of the CPUSCR during the following steps cause interrupt to be disabled (clearing MSR_{EE} and MSR_{CE}). This will ensure that external interrupt sources do not cause single-step errors.

To single-step the CPU:

- debugger scans in either a new or a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Noexit OnCE Command value.
- The debugger scans out the OSR with "no-register selected", Go cleared, and determines that the PCU has re-entered the Debug state and that no ERR condition occurred

To return the CPU to normal operation (without disabling external debug mode)

- The OCR_{DR} control bit should be cleared, leaving the OCR_{WKUP} bit set
- The debugger restores the CPUSCR with a previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared

To exit External Debug Mode

- The debugger should place the CPU in the debug state via the OCR_{DR} with OCR_{WKUP} asserted, scanning out and saving the CPUSCR
- The debugger should write the DBCR0–8 registers as needed, likely clearing every enable except the $EDBCR0_{EDM}$ bit
- The debugger should write the DBSR to a cleared state
- The debugger should re-write the DBCR0 with all bits including EDM cleared

- The debugger should clear the OCR_{DR} bit
- The debugger restores the CPUSCR with the previously saved value of the CPUSCR (with appropriate modification of the PC and IR as described in Control State Register (CTL)), with a Go+Exit OnCE Command value.
- The OCR_{WKUP} bit may then be cleared

Note

These steps are meant by way of examples, and are not meant to be an exact template for debugger operation.

Chapter 66
Z2 Nexus 3 Module

66.1 Introduction

The e200z210n3Nexus 3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port in these processors.

66.1.1 General description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

66.1.2 Terms and definitions

The following table contains a set of terms and definitions associated with the Nexus 3 module.

Table 66-1. Terms and definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design. World wide Web documentation at http://www.ieee-isto.org/Nexus5001
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	"System-on-a-Chip". SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase "according to the standard" is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

66.1.3 Feature list

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

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- Auxiliary interface for higher data input
    - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
    - One (1) or two (2) Message Start/End Out pins (**nex_mseo_b[1:0]**)
    - One (1) Read/Write Ready pin (**nex_rdy_b**) pin
    - One (1) Read/Write Error pin (**nex_err_b**) pin
    - One (1) Watchpoint Event output pin (**evto_b**)
    - One (1) Event In pin (**nex_evti_b**)
    - One (1) MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via SoC signaling input (**nex_sfwcntl_en**)
- Indicates to the FCCU whether it becomes active during functional mode.

Note

For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (at the SoC level). For single Nexus implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM - maximum number of MDO pins) or Reduced Port Mode (RPM - minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note

The configuration of the Message Start/End Out pins (1 or 2) is determined at the SOC integration level. This option will be hard-wired based on SOC bandwidth requirements.

66.1.4 Functional block diagram

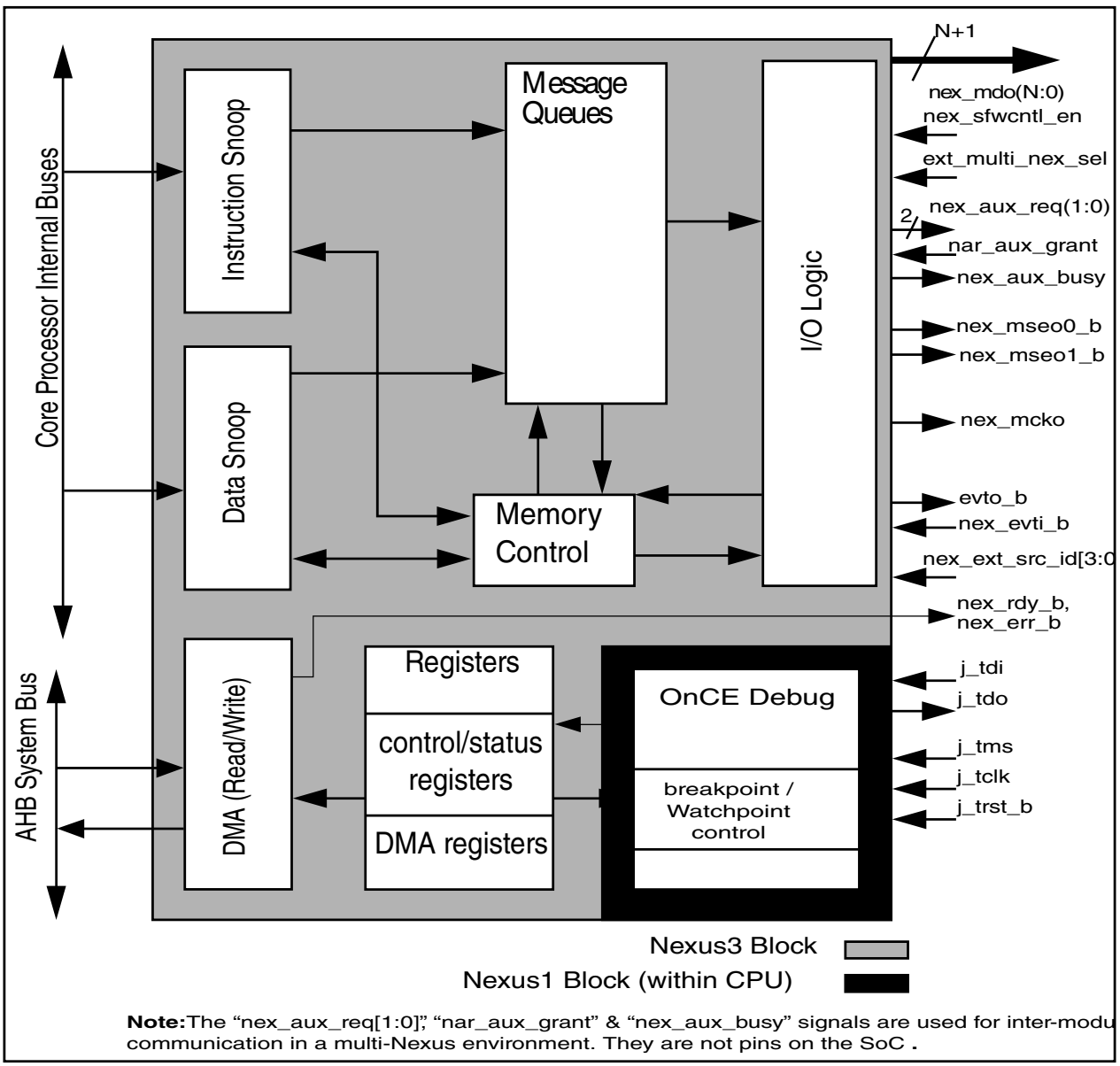


Figure 66-1. Nexus 3 functional block diagram

66.2 Enabling Nexus 3 operation

The Nexus module is enabled by loading a single instruction (*NEXUS3-ACCESS*) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the nex_evti_b input signal is asserted at the time that j_trst_b is initially negated, or is asserted during the Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test_Logic_Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

66.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

66.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001-2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in Table 66-2.

Table 66-2. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 66-7)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)

Table continues on the next page...

Table 66-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 66-7)
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from $DEVENT_{DQTAG}$ register field
	1	32	DQDATA	variable	exported data taken from DDAM register
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch Message w/Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)

Table continues on the next page...

Table 66-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 13
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 66-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Data Trace - Data Read Message w/ Sync	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 66-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to Table 66-5) - indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (See Section Resource Full Messages.)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions

Table continues on the next page...

Table 66-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	32	HIST	variable	branch / predicate instruction history (see Branch Trace Messaging types)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Indirect Branch History Message w/ Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (See Branch Trace Messaging types.)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)
Program Trace - Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated w/ program flow (Refer to Table 66-6)
	2	2	CDF	fixed	# fields of information in CDATA. 00 - reserved, 01 - one field (CDATA1) (reserved), 10 - two fields (CDATA1 + CDATA2), 11 - three fields (reserved)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 - [branch / predicate instruction history] (See Program Correlation Messages.)
	0	32	CDATA2	variable	correlation data field 2- PID info (See Program Correlation Messages.)
	0	30	TSTAMP	variable	Timestamp value (optionally generated field)

Note

Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/ Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in Branch Trace Messaging types. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 66-3 shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 66-3. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (Debug Status messages, etc.)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

Table 66-4 shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 66-4. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 - 1111	Reserved

Table 66-5 shows the encodings used for resource codes for certain messages.

Table 66-5. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

Table 66-6 shows the event code encodings used for certain messages.

Table 66-6. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010-0011	Reserved for future functionality
0100	Disabling Program Trace

Table continues on the next page...

Table 66-6. Event Code (EVCODE) Encoding (TCODE = 33) (continued)

Event Code	Description
0101	Process ID value is established in PID0/NPIDR via **mtspr PID0/NPIDR**
0110-1000	Reserved for future functionality
1001	Begin masking of program trace messages due to MSR_{PMM}=0 and $DC4_{PTMARK}$=1
1010	Branch and link occurrence (direct branch function call)
1011-1111	Reserved for future functionality

Table 66-7 shows the data trace size encodings used for certain messages.

Table 66-7. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 - no data
0001	Byte
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001-1111	Reserved

66.4 Nexus 3 Programmer's model

This section describes the Nexus 3 programmer's model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See Nexus 3 Register Access via JTAG/OnCE and Nexus 3 Register Access via Software for details on Nexus 3 register access.

Note

Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

The following table details the register map for the Nexus 3 module.

Table 66-8. Nexus 3 register map

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #[1]
Client Select Control (CSC)[2]	0x1	R	0x02	-	
Port Configuration Register (PCR)[3]	PCR_INDEX[2]	R/W	-	-	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	-
Reserved	0x8	R/W	0x18	0x19	
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	-
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	-
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 -> 0x2F	-	0x28->0x5E	0x29->5F	
Development Status (DS)	0x30	R	0x60	-	409
Reserved	0x31	R/W	0x62	0x63	
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	-	0x68	0x69	
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 -> 0x3F	-	0x72->0x7E	0x73->7F	

1. Software access via the **mfdcr** and **mtdcr** instructions use these values for the DCR number. Software writes to these registers via **mtdcr** when **nex_sfwcntl_en** is negated are ignored.
2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.

3. The "PCR_INDEX" is a parameter determined by the SoC.

66.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.

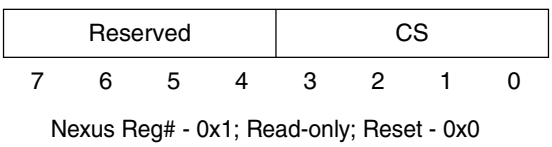


Figure 66-2. Client Select Control (CSC) register

Table 66-9. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0xx - Nexus client (SoC level)

66.4.2 Port Configuration Register (PCR) - reference only

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.

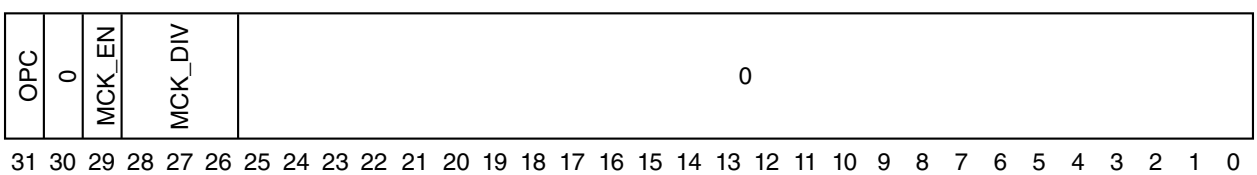


Figure 66-3. Port Configuration Register

Table 66-10. PCR field descriptions

Bit	Name	Description
31	OPC	Output Port Mode Control (SoC Level) 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined by SOC) 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined by SOC)
30	—	Reserved for future functionality

Table continues on the next page...

Table 66-10. PCR field descriptions (continued)

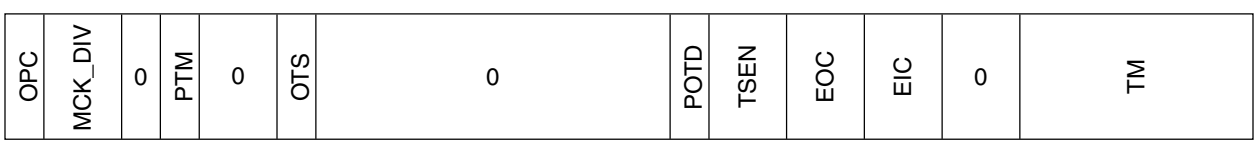
Bit	Name	Description
29	MCK_EN	MCKO Clock Enable (SoC Level) 0 **nex_mcko** is disabled 1 **nex_mcko** is enabled
28:26	MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level) 000 **nex_mcko** is 1x processor clock freq. 001 **nex_mcko** is 1/2x processor clock freq. 010 Reserved (default to 1/2x processor clock freq.) 011 **nex_mcko** is 1/4x processor clock freq. 100 Reserved (default to 1/2x processor clock freq.) 101 Reserved (default to 1/2x processor clock freq.) 110 Reserved (default to 1/2x processor clock freq.) 111 **nex_mcko** is 1/8x processor clock freq.
25:0	—	Reserved for future functionality

Note

The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

66.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in Figure 66-4 and its fields are described in Table 66-11.



Nexus Reg# - 0x2; DCR - 368; Read/Write; Reset - 0x0

Figure 66-4. Development Control Register 1

Table 66-11. DC1 field descriptions

Bits	Name	Description
31	OPC	Output Port Mode Control 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined
30:29	MCK_DIV	MCKO Clock Divide Ratio (see note below) 00 **nex_mcko** is 1x processor clock freq. 01 **nex_mcko** is 1/2x processor clock freq. 10 **nex_mcko** is 1/4x processor clock freq. 11 **nex_mcko** is 1/8x processor clock freq.
28	—	Reserved for future functionality
27	PTM	Program Trace Method 0 Program Trace uses traditional Branch Messages 1 Program Trace uses Branch History Messages
26:25	—	Reserved for future functionality
24	OTS	Ownership Trace PID Select 0 PID0 data is transmitted within Ownership Trace Messages 1 Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality
14	POTD	Periodic Ownership Trace Disable 0 Periodic Ownership Trace message events are enabled 1 Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable - (not implemented, write to 00) 00 Timestamp is disabled
11:10	EOC	EVTO Control 00 **evto_b** upon occurrence of Watchpoints (configured in DC2 and DC3) 01 **evto_b** upon entry into Debug Mode 1x Reserved
9:8	EIC	EVTI Control 00 **nex_evti_b** is used for synchronization (Program Trace Sync msg is generated) 01 **nex_evti_b** is used for Debug request 10 **nex_evti_b** is disabled 1X Reserved
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode[1] 000000 All Trace Disabled XXXXX1 Ownership Trace enabled XXXX1X Data Trace enabled XXX1XX Program Trace enabled XX1XXX Watchpoint Trace enabled X1XXXX Reserved 1XXXXX Data Acquisition Trace enabled

1. This field may be updated by hardware in response to watchpoint triggering if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering.

Note

The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) MUST ONLY be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

66.4.4 Nexus Development Control Registers 2 & 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Core (e200z210n3) Core Debug Support chapter.

Development Control Register 2 is shown in Figure 66-5 and its fields are described in Table 66-12.

0	0	0	0	EWC

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x3; DCR - 369; Read/Write; Reset - 0x0

Figure 66-5. Development Control Register 2

Table 66-12. DC2 field descriptions

Bits	Name	Description
31:16	—	Reserved
15:0	EWC	EVTO Watchpoint Configuration[1] 0000000000000000 No Watchpoints #0-15 trigger **evto_b** XXXXXXXXXXXXXXX1 Watchpoint #0 triggers **evto_b** XXXXXXXXXXXXXX1X Watchpoint #1 triggers **evto_b** XXXXXXXXXXXXX1XX Watchpoint #2 triggers **evto_b** XXXXXXXXXXXX1XXX Watchpoint #3 triggers **evto_b** XXXXXXXXXXX1XXXX Watchpoint #4 triggers **evto_b** XXXXXXXXXX1XXXXX Watchpoint #5 triggers **evto_b** XXXXXXXXX1XXXXXX Watchpoint #6 triggers **evto_b** XXXXXXXX1XXXXXXX Watchpoint #7 triggers **evto_b**

Table 66-12. DC2 field descriptions

Bits	Name	Description
		XXXXXXX1XXXXXXXX Watchpoint #8 triggers **evto_b** XXXXXX1XXXXXXXXX Watchpoint #9 triggers **evto_b** XXXXX1XXXXXXXXXX Watchpoint #10 triggers **evto_b** XXXX1XXXXXXXXXXX Watchpoint #11 triggers **evto_b** XXX1XXXXXXXXXXXX Watchpoint #12 triggers **evto_b** XX1XXXXXXXXXXXXX Watchpoint #13 triggers **evto_b** X1XXXXXXXXXXXXXX Watchpoint #14 triggers **evto_b** 1XXXXXXXXXXXXXXX Watchpoint #15 triggers **evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in Figure 66-6 and its fields are described in Table 66-13.

0	0	0	0	0	EWC

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x4; DCR - 370; Read/Write; Reset - 0x0

Figure 66-6. Development Control 3 (DCR3) register

Table 66-13. DC3 field descriptions

Bits	Name	Description
31:16	—	Reserved
15:14	—	Reserved for watchpoint expansion
13:0	EWC	EVTO Watchpoint Configuration[1] 00000000000000 No Watchpoints #16-#29 trigger **evto_b** XXXXXXXXXXXXX1 Watchpoint #16 triggers **evto_b** XXXXXXXXXXXX1X Watchpoint #17 triggers **evto_b** XXXXXXXXXXX1XX Watchpoint #18 triggers **evto_b** XXXXXXXXXX1XXX Watchpoint #19 triggers **evto_b** XXXXXXXXX1XXXX Watchpoint #20 triggers **evto_b** XXXXXXXX1XXXXX Watchpoint #21 triggers **evto_b** XXXXXXX1XXXXXX Watchpoint #22 triggers **evto_b** XXXXXX1XXXXXXX Watchpoint #23 triggers **evto_b** XXXXX1XXXXXXXX Watchpoint #24 triggers **evto_b** XXXX1XXXXXXXXX Watchpoint #25 triggers **evto_b** XXX1XXXXXXXXXX Watchpoint #26 triggers **evto_b** XX1XXXXXXXXXXX Watchpoint #27 triggers **evto_b**

Table 66-13. DC3 field descriptions

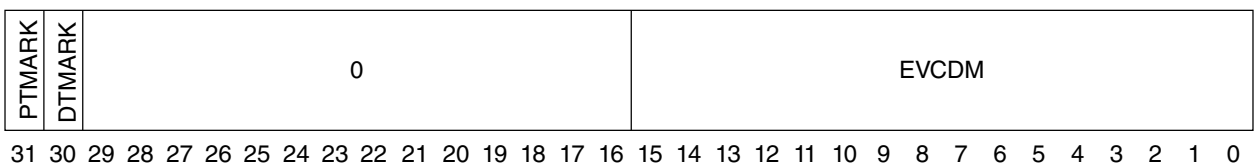
Bits	Name	Description
		X1XXXXXXXXXXX Watchpoint #28 triggers **evto_b** 1XXXXXXXXXXXX Watchpoint #29 triggers **evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

66.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in Figure 66-7 and its fields are described in Table 66-14.



Nexus Reg# - 0x5; DCR - 371; Read/Write; Reset - 0x0

Figure 66-7. Development Control 4 (DC4) register

Table 66-14. DC4 field descriptions

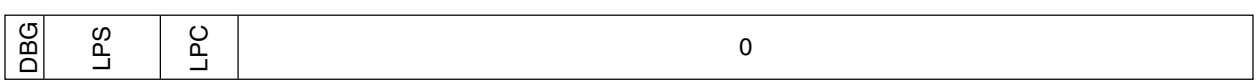
Bits	Name	Description
31	PTMARK	Program Trace Mark 0 Ignore MSR_{PMM} for masking program trace messages 1 Mask program trace messages when MSR_{PMM}='0', unmask program trace messages when MSR_{PMM}='1'
30	DTMARK	DTMARK Data Trace Mark 0 Ignore MSR_{PMM} for masking data trace messages 1 Mask data trace messages when MSR_{PMM}='0', unmask data trace messages when MSR_{PMM}='1'
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask For implemented EVCODEs. please see Table 66-6. 0000000000000000 No EVCODEs masked for Program Correlation Messages XXXXXXXXXXXXXXX1 EVCODE #0 is masked for Program Correlation Messages XXXXXXXXXXXXXX1X EVCODE #1 is masked for Program Correlation Messages XXXXXXXXXXXXX1XX EVCODE #2 is masked for Program Correlation Messages XXXXXXXXXXXX1XXX EVCODE #3 is masked for Program Correlation Messages

Table 66-14. DC4 field descriptions

Bits	Name	Description
		XXXXXXXXXXX1XXXX EVCODE #4 is masked for Program Correlation Messages
		XXXXXXXXXX1XXXXX EVCODE #5 is masked for Program Correlation Messages
		XXXXXXXXX1XXXXXX EVCODE #6 is masked for Program Correlation Messages
		XXXXXXXX1XXXXXXX EVCODE #7 is masked for Program Correlation Messages
		XXXXXXX1XXXXXXXX EVCODE #8 is masked for Program Correlation Messages
		XXXXXX1XXXXXXXXX EVCODE #9 is masked for Program Correlation Messages
		XXXXX1XXXXXXXXXX EVCODE #10 is masked for Program Correlation Messages
		XXXX1XXXXXXXXXXX EVCODE #11 is masked for Program Correlation Messages
		XXX1XXXXXXXXXXXX EVCODE #12 is masked for Program Correlation Messages
		XX1XXXXXXXXXXXXX EVCODE #13 is masked for Program Correlation Messages
		X1XXXXXXXXXXXXXX EVCODE #14 is masked for Program Correlation Messages
		1XXXXXXXXXXXXXXX EVCODE #15 is masked for Program Correlation Messages

66.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.



Nexus Reg# - 0x4; DCR - 409; Read-only; Reset - 0x0

Figure 66-8. Development Status Register

Table 66-15. DS field description

Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (**jd_debug_b** signal asserted)
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting state (**p_waiting** signal asserted) 010 Halted State (**p_halted** signal asserted) 011 Reserved

Table continues on the next page...

Table 66-15. DS field description (continued)

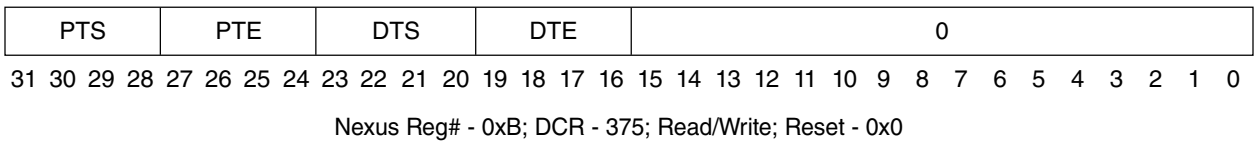
Bits	Name	Description
		1XX Reserved
27:26	LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (**p_halted** signal asserted) 10 CPU in Stopped state (**p_stopped** signal asserted) 11 CPU in Waiting state (**p_waiting** signal asserted)
25:0	—	Reserved for future functionality (read as 0)

66.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related "window" for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4'b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take

precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Core (e200z210n3) Core Debug Support chapter.



Nexus Reg# - 0xB; DCR - 375; Read/Write; Reset - 0x0

Figure 66-9. Watchpoint Trigger (WT) Register

Table 66-16 details the Watchpoint Trigger register fields.

Table 66-16. WT field descriptions

Bits	Name	Description
31:28	PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register
27:24	PTE	PTE - Program Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0

Table continues on the next page...

Table 66-16. WT field descriptions (continued)

Bits	Name	Description
		0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.

PTST																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x35; DCR - 412; Read/Write; Reset - 0x0

Figure 66-10. Program Trace Start Trigger Control (PTSTC) register

Table 66-17 details the PTSTC register fields.

Table 66-17. Program Trace Start Trigger Control Register Fields

Bits	Name	Description
31:0	PTST	Program Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16

Table 66-17. Program Trace Start Trigger Control Register Fields

Bits	Name	Description
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		XXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		XX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #29
		X1XXXXXXXXXXXXXXXXXXX Use Watchpoint #30
		1XXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.

PTET

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x36; DCR - 413; Read/Write; Reset - 0x0

Figure 66-11. Program Trace End Trigger Control (PTETC) register

Table 66-18 details the PTETC register fields.

Table 66-18. PTETC field descriptions

Bits	Name	Description
31:0	PTET	PTET - Program Trace End Trigger Control
		00000000000000000000000000000000 Trigger disabled
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7

Table 66-18. PTETC field descriptions

Bits	Name	Description
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #30
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

	DTST

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x37; DCR - 414; Read/Write; Reset - 0x0

Figure 66-12. Data Trace Start Trigger Control (DTSTC) register

The following table details the DTSTC register fields.

Table 66-19. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)

Table continues on the next page...

Table 66-19. DTSTC field descriptions (continued)

Bits	Name	Description
29:0	DTST	Data Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18 XXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19 XxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20 XXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21 XXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22 XXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23 XXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24 XXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25 XXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26 XX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27 X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28 1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

For extended Data Trace end trigger control, the DTETC register is used.

0	DTET

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 038; DCR - 415; Read/Write; Reset - 0x0

Figure 66-13. Data Trace End Trigger Control (DTETC) register

The following table details the DTETC register fields.

Table 66-20. DTET field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTET	Data Trace End Trigger Control 00000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18 XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19 XxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20 XXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21 XXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22 XXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23 XXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24 XXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25 XX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26 X1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27

Table 66-20. DTET field descriptions

Bits	Name	Description
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28 1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

66.4.8 Nexus Watchpoint Mask (WMSK) register

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages ($DC1_{TM}$ must also be programmed to generate Watchpoint Trace Messages).

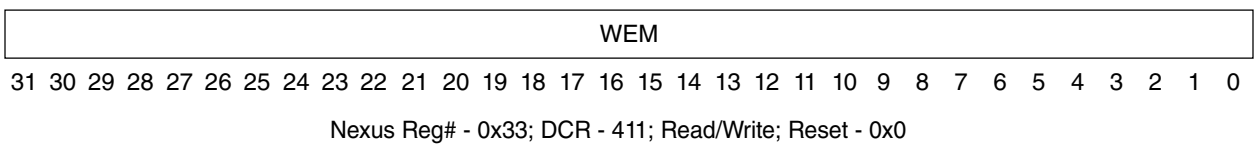


Figure 66-14. Watchpoint Mask Register

The following table details the Watchpoint Trigger register fields.

Table 66-21. WMSK field descriptions

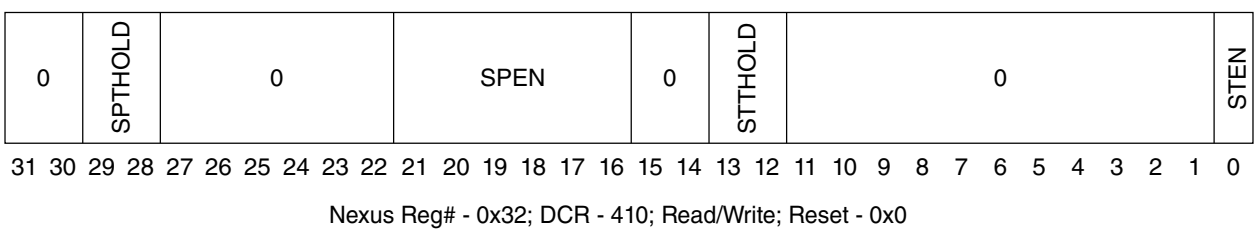
Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging 00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Watchpoint #0 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Watchpoint #1 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Watchpoint #2 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Watchpoint #3 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Watchpoint #4 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Watchpoint #5 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Watchpoint #6 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Watchpoint #7 enabled for WTM XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Watchpoint #8 enabled for WTM XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Watchpoint #9 enabled for WTM XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Watchpoint #10 enabled for WTM XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Watchpoint #11 enabled for WTM XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Watchpoint #12 enabled for WTM XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Watchpoint #13 enabled for WTM XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Watchpoint #14 enabled for WTM XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Watchpoint #15 enabled for WTM

Table 66-21. WMSK field descriptions

Bits	Name	Description
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Watchpoint #16 enabled for WTM
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Watchpoint #17 enabled for WTM
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Watchpoint #18 enabled for WTM
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Watchpoint #19 enabled for WTM
		XXXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Watchpoint #20 enabled for WTM
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Watchpoint #21 enabled for WTM
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Watchpoint #22 enabled for WTM
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Watchpoint #23 enabled for WTM
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #24 enabled for WTM
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #25 enabled for WTM
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #26 enabled for WTM
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #27 enabled for WTM
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #28 enabled for WTM
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #29 enabled for WTM
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #30 enabled for WTM
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #31 enabled for WTM

66.4.9 Nexus Overrun Control Register (OVCR)

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.



Nexus Reg# - 0x32; DCR - 410; Read/Write; Reset - 0x0

Figure 66-15. Nexus Overrun Control Register

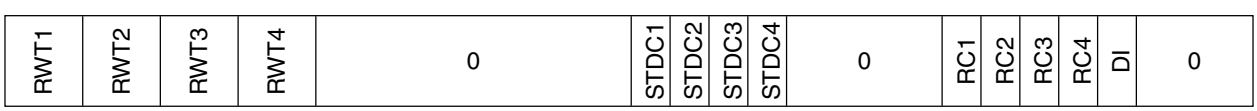
Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	SPTHOLD	Suppression Threshold 00 - Suppression threshold is when message queues are 1/4 full 01 - Suppression threshold is when message queues are 1/2 full 10 - Suppression threshold is when message queues are 3/4 full 11 - Reserved
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 - Suppression is disabled xxxxx1 - Ownership Trace message suppression is enabled xxxx1x - Data Trace message suppression is enabled xxx1xx - Program Trace message suppression is enabled xx1xxx - Watchpoint Trace message suppression is enabled x1xxxx - Reserved 1xxxxx - Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	STTHOLD	Stall Threshold 00 - Stall threshold is when message queues are 1/4 full 01 - Stall threshold is when message queues are 1/2 full 10 - Stall threshold is when message queues are 3/4 full 11 - Reserved
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 - Stalling is disabled 1 - Stalling is enabled

Figure 66-16. Nexus Overrun Control Register Fields

66.4.10 Data Trace Control Register (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.



Nexus Reg# - 0xD; DCR - 376; Read/Write; Reset - 0x0

Figure 66-17. Data Trace Control Register

The following table details the Data Trace Control register fields.

Table 66-22. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)
15	STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1 1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
12	STDC4	Stack Trace Disable Control 4 0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1 0 Condition trace on address within range

Table continues on the next page...

Table 66-22. DTC field descriptions (continued)

Bits	Name	Description
		1 Condition trace on address outside of range
6	RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5	RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range
4	RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module. The setting of the DI bit also affects the information provided on the data trace port outputs (See the "Data Trace Port Signals" section in the Core (e200z210n3) Core Debug Support chapter.)
2:0	—	Reserved for future functionality (read as 0)

66.4.11 Data Trace Start Address Registers (DTSA1-4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0xE; DCR - 377; Read/Write; Reset - 0x0

Figure 66-18. Data Trace Start Address 1 (DTSA1) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0xF; DCR - 378; Read/Write; Reset - 0x0

Figure 66-19. Data Trace Start Address 2 (DTSA2) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x10; DCR - 379; Read/Write; Reset - 0x0

Figure 66-20. Data Trace Start Address 3 (DTSA3) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x11; DCR - 380; Read/Write; Reset - 0x0

Figure 66-21. Data Trace Start Address 4 (DTSA4) register

66.4.12 Data Trace End Address (DTEA1-4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x12; DCR - 381; Read/Write; Reset - 0x0

Figure 66-22. Data Trace End Address 1 (DTEA1) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x13; DCR - 382; Read/Write; Reset - 0x0

Figure 66-23. Data Trace End Address 2 (DTEA2) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x14; DCR - 383; Read/Write; Reset - 0x0

Figure 66-24. Data Trace End Address 3 (DTEA3) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x15; DCR - 408; Read/Write; Reset - 0x0

Figure 66-25. Data Trace End Address 4 (DTEA4) register

Table 66-23 illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 66-23. Data trace - address range options

Programmed values	Range Control Bit Value	Range Selected	
DTSA < DTEA	0	DTSA ->	<- DTEA
DTSA < DTEA	1	<- DTSA	DTEA ->
DTSA > DTEA	N/A	Invalid Range - no trace	
DTSA = DTEA	N/A	Invalid Range - no trace	

Note

DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are *inclusive* of the DTSA and DTEA addresses for Range Control settings indicating "within range," and are *exclusive* of the DTSA and DTEA addresses for Range Control settings indicating "outside of range."

Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when DTC_{DI}=1, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (i.e. when the instruction prefetch is discarded).

66.4.13 Read/Write Access Control/Status (RWCS) register

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes.

Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values.

The RWCS Register also provides Read/Write Access Status information per Table 66-25.

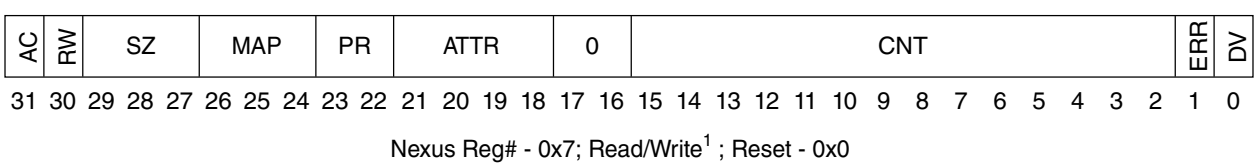


Figure 66-26. Read/Write Access Control/Status (RWCS) register

NOTES:

[1]ERR and DV are read-only

Table 66-24. RWCS field description

Bits	Name	Description
RWCS[31]	AC	Access Control 0 End access/ Access has completed 1 Start access
RWCS[30]	RW	Read/Write Select 0 Read access 1 Write access
RWCS[29:27]	SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 100 - 111 Reserved (default to word)
RWCS[26:24]	MAP	MAP Select 000 Primary memory map 001-111 Reserved
RWCS[23:22]	PR[1]	Read/Write Access Priority 00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority
RWCS[21:18]	ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx **p_d_hprot[4]** driven to 0 for accesses x1xx **p_d_hprot[4]** driven to 1 for accesses xx0x **p_d_hprot[3]** driven to 0 for accesses xx1x **p_d_hprot[3]** driven to 1 for accesses xxx0 **p_d_hprot[2]** driven to 0 for accesses xxx1 **p_d_hprot[2]** driven to 1 for accesses
RWCS[17:16]	—	RES - Reserved for future functionality
RWCS[15:2]	CNT	Access Control Count hhhh Number of accesses of word size SZ
RWCS[1]	ERR[2]	Read/Write Access Error (see Table 66-25)
RWCS[0]	DV[2]	Read/Write Access Data Valid (see Table 66-25)

1. The priority functionality is not currently implemented.
2. ERR and DV are read-only.

Table 66-25. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

66.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.

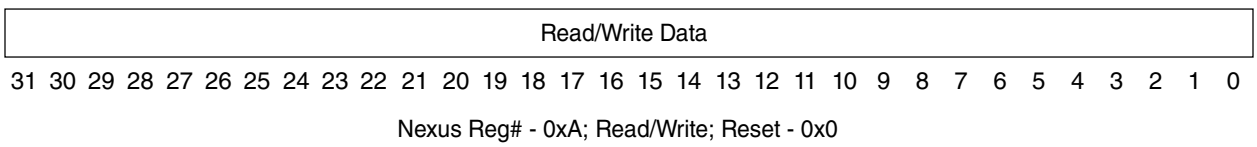


Figure 66-27. Read/Write Access Data (RWD) register

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. Table 66-26 shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 66-26. RWD data placement for ransfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	-	-	-	X
Half	x x 0	0 0 1	-	-	X	X
Word	x 0 0	0 1 0	X	X	X	X
Doubleword	0 0 0	0 1 1				
first RWD pass (low order data)			X	X	X	X
second RWD pass (high order data)			X	X	X	X

Table Notes:

"X" indicates byte lanes with valid data

"-" indicates byte lanes that will contain unused data.

Table 66-27 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 66-27. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	-	-	-	AHB[7:0]
Byte @001	0 0 1	-	-	-	AHB[15:8]
Byte @010	0 1 0	-	-	-	AHB[23:16]
Byte @011	0 1 1	-	-	-	AHB[31:24]
Byte @100	1 0 0	-	-	-	AHB[39:32]
Byte @101	1 0 1	-	-	-	AHB[47:40]
Byte @110	1 1 0	-	-	-	AHB[55:48]
Byte @111	1 1 1	-	-	-	AHB[63:56]
Half @000	0 0 0	-	-	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	-	-	AHB[31:24]	AHB[23:16]
Half @100	1 0 0	-	-	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	-	-	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]
Doubleword @000	0 0 0				
first RWD pass		AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
second RWD pass		AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]

Table Notes:

"-" indicates byte lanes that will contain unused data.

66.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

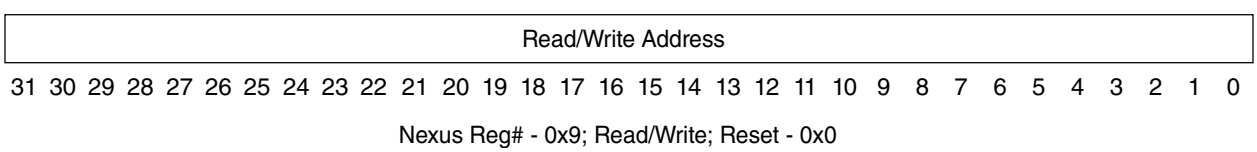


Figure 66-28. Read/Write Access Address (RWA) register

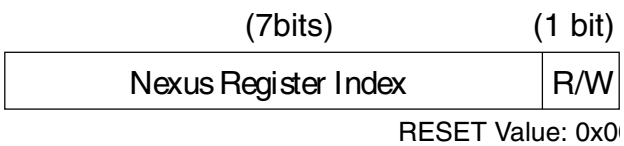
66.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction ("*NEXUS3-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in Table 66-8.

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See IEEE 1149.1 (JTAG) RD/WR sequences.)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see Table 66-8), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:



Nexus Register Index	Selected from values in Table 66-8
Read/Write (R/W):	0 - Read 1 - Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
   a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the "Capture-DR" state.
   b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the "Update-DR" state.

66.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these access only occur when no

possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. Table 66-8 shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

66.7 Nexus message fields

Nexus messages are composed of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

66.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

66.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

66.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant '1'. To recreate the original address, the relative address is XORed with the previously decoded address.

The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365

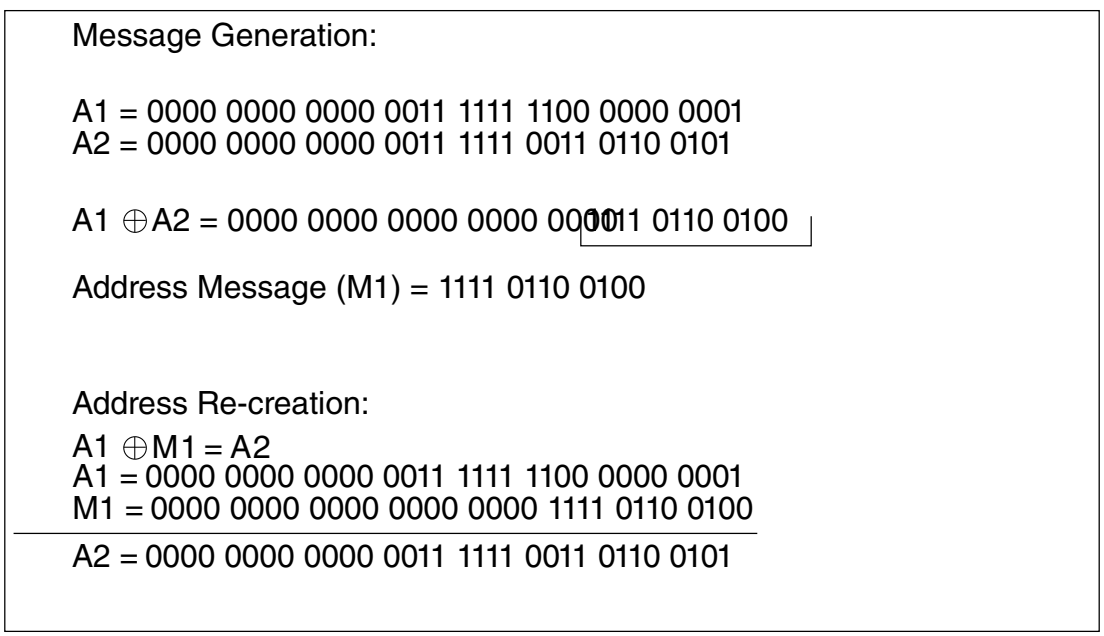


Figure 66-29. Relative address generation and recreation

66.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

66.7.5 Timestamp field (TSTAMP)

Timestamps may be optionally applied to messages. Enabling of timestamping is controlled by the nex_tsen input signal. The timestamping mode (timestamp frequency of occurrence) is controlled by the value of nex_tsmode[0:1] and the timestamp value is based on the nex_timebase[0:29] inputs. These signals are first synchronized to the nex_clk clock domain, and for nex_timebase[0:29], a gray-code to binary value conversion is performed. The timestamp value is applied to messages as they exit the message queues. Leading zeros in the timestamp value are removed. The table shows the nex_tsmode[0:1] encodings used for the timestamp mode.

Table 66-28. nex_tsmode[0:1] timestamp mode encoding

nex_tsmode[0:1]	Timestamp Mode
00	Timestamp applied to every message
01	Timestamp applied to every 4th message
10	Timestamp applied to every 16th message
11	Timestamp applied to every 64th message

66.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

66.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

66.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to Nexus Overrun Control Register (OVCR) for complete programming options.

66.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in

Message Queue Overrun. Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

66.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstances:

- More than three messages are generated in the same cycle

Table 66-29 lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages that did not successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in Table 66-29.

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM ->DQM -> PCM_{PIDMSG} -> OTM -> BTM -> DTM)-> Instruction1 (newer instruction) (WPM -> DQM -> OTM -> BTM -> DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 66-29. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A[1]
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A[1]
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM - PID/NPIDR update (Program Correlation Message)	2	OTM Error Message

Table continues on the next page...

Table 66-29. Message Type Priority and Message Dropped Responses (continued)

Message Type	Message	Priority	Message Dropped Response
OT (Ownership)	OTM - PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message[2]
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM - Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

66.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

66.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

66.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

66.8.8 Program Trace Sync Message Priority Loss Response

If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

66.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

66.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:

(8 bits)	(4 bits)	(6 bits)
DS[31:24]	Src. Proc.	TCODE (000000)

Fixed length = 18 bits

Figure 66-30. Debug Status message format

66.10 Error messages

Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note

The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see Table 66-3 and Table 66-4):

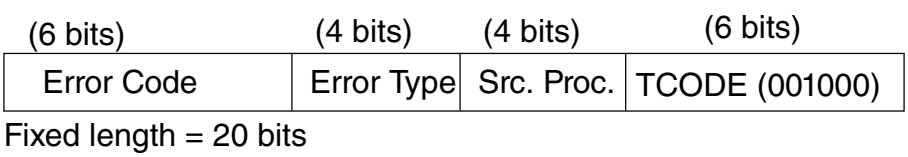


Figure 66-31. Error message format

66.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

66.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

66.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Core (e200z210n3) processors contain a *Power ISA 2.06* defined "Process ID" register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr/mtspr** instructions.

Note

The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr/mtspr** instructions. This register is accessible in user or supervisor mode.

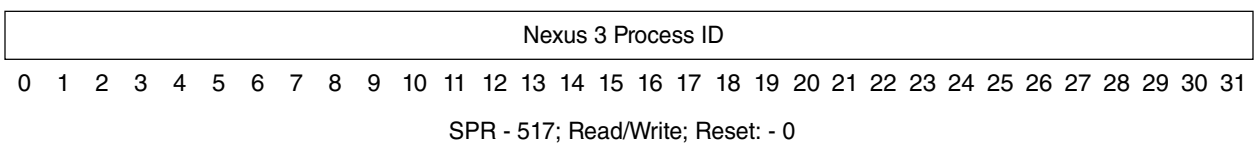


Figure 66-32. Nexus 3 Process ID Register (NPIDR)

Note

OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

The process ID source (PID0 or NPIDR) is selected by the setting of the $DC1_{OTS}$ control bit.

There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and $DC1_{OTS}$ indicates PID0 should be used, or when the NPIDR register is written to by the processor and $DC1_{OTS}$ indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for

PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.

- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting $DC1_{POTD}$.

Ownership trace information is messaged out in the following format:

(1-32 bits)	(4 bits)	(4 bits)	(6 bits)
Process ID	PID Index (0000)	Src. Proc.	TCODE (000010)

Variable length = 15-46 bits

Figure 66-33. Ownership Trace Message format

66.12 Program trace

This section details the program trace mechanism supported by Nexus3 for the e200z210n3 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for Core (e200z210n3) processors is accomplished by snooping the internal address bus, attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

66.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

66.12.1.1 Indirect Branch Message instructions

Table 66-30 shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 66-30. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**bcctr, bcctrl, bclr, bclrl, se_bctr, se_bctrl, se_blr, se_blrl**
System Call / Trap exceptions taken	**se_sc, tw**
Return from interrupts / exceptions	**se_rfi, se_rfci, se_rfdi**
Exit from reset with Program Trace Enabled	**Indirect branch with Sync, target address is initial instruction, count = 1**

66.12.1.2 Direct Branch Message Instructions

Table 66-31 shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 66-31. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**b, ba, bl, bla, bc, bca, bcl, bcla, se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

66.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken.

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

66.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of "events" that occurred with respect to direct branches simply by the order in which messages are sent out.

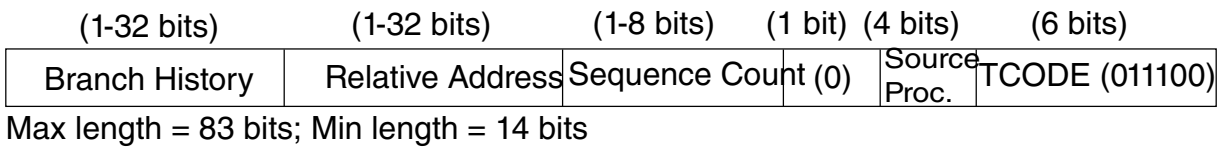
Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

66.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages - Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages - Indirect Branch History, and Indirect Branch History with Synchronization Messages.

66.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If $DC1_{PTM}$ is set to '1', indirect branch information is messaged out in the following format:

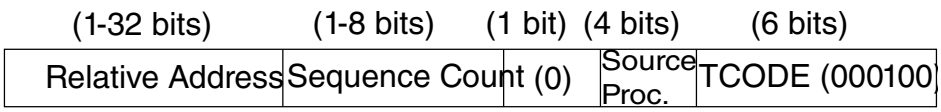


Max length = 83 bits; Min length = 14 bits

Figure 66-34. Indirect branch message (history) format

66.12.2.2 Indirect branch messages (traditional)

If $DC1_{PTM}$ is cleared to '0', indirect branch information is messaged out in the following format:



Max length = 51 bits; Min length = 13 bits

Figure 66-35. Indirect branch message format

66.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:

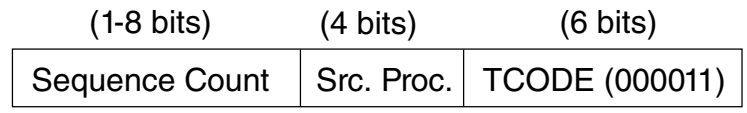


Figure 66-36. Direct Branch message format

Note

When $DC1_{PTM}$ is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

66.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

66.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that trigger message generation are included in the ICNT.

For Branch History messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions that trigger message generation are included in the ICNT. Instructions that generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a w/sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

66.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 66-32. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	**bcctr, bcctrl, bclr, bclrl**
Not taken direct branches	0	**b, ba, bc, bca, bla, bcla, bl, bcl**
Taken direct branches	1	**b, ba, bc, bca, bla, bcla, bl, bcl**[1]

1. If the EVCODE for direct branch function calls is not masked in DC4, taken **bl** and **bcl** instructions will generate Program Correlation Messages and will not be logged in the history buffer.1

66.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

66.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/ predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow, or, if program trace is in history mode, since the last instruction that recorded history information.

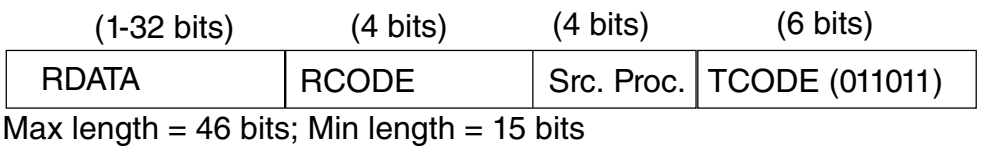


Figure 66-37. Resource Full Message format

The following table shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 66-33. RCODE encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch HIstory. This type of packet is terminated by a stop bit set to 1 after the last history bit.

66.12.5 Program Correlation Messages

Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a "Branch and Link" instruction executes (direct branch function call — **bl/bcl/bla/bcla**-type instructions)
- When program trace becomes masked due to MSR_{PMM}='0' and $DC4_{PTMARK}$='1'.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to Table 66-6 for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:

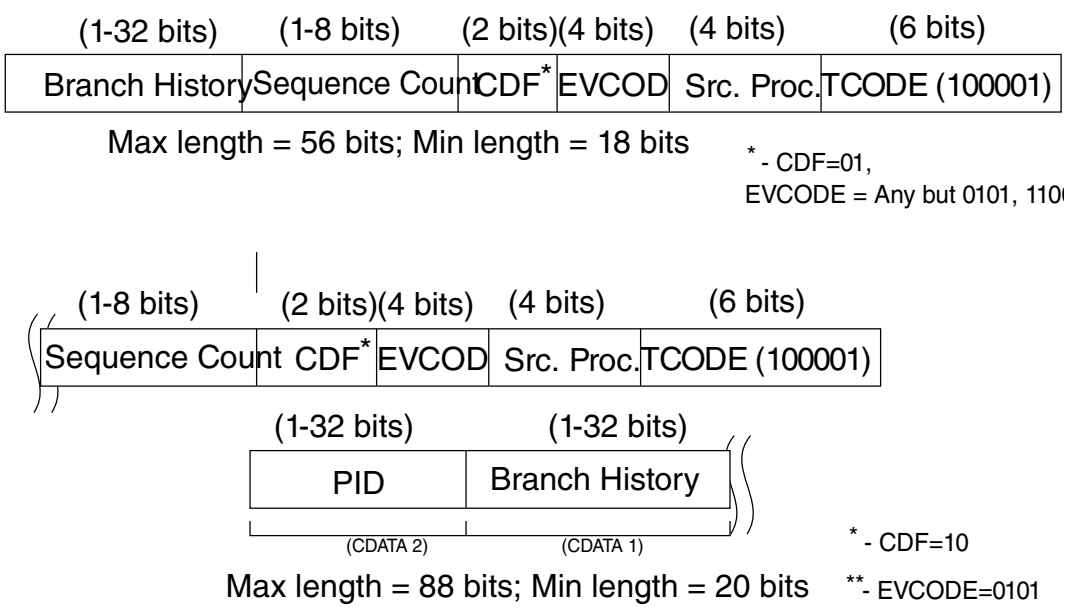


Figure 66-38. Program Correlation message formats

If timestamping is enalbed, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

66.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

66.12.6 Program Trace Overflow Error messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

66.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 66-34):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.

- Upon occurrence of queue overrun (can be caused by any trace message) or suppression, provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to $MSR_{PMM} \rightarrow$ '1' with $DC4_{PTMARK}$='1'.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Seq. Count (0)	(0)	Source Proc.	TCODE (001001)

Max length = 51 bits; Min length = 13 bits

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

Exception conditions that result in Program Trace Synchronization are summarized in Table 66-34.

Table 66-34. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (**p_reset_b**), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address of the instruction that caused program trace to be enabled, or may be the address of the first instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

66.12.8 Program Trace Direct/Indirect Branch with Sync messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 66-34):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync Messages is as follows:

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (001011 or 001100)

Max length = 51 bits; Min length = 13 bits

Figure 66-39. Direct/Indirect Branch with Sync message format

The format for Program Trace Indirect Branch History with Sync. messages is as follows:

(1-32 bits)	(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Branch History	Full Target Address	Sequence Count	(0)	Source Proc.	TCODE (011101)

Max length = 83 bits; Min length = 14 bits

Figure 66-40. Indirect Branch History w/ Sync. message format

Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in Table 66-35.

Table 66-35. Program Trace Exception Summary for w/Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync.Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the *next* Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM -> DQM -> PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

66.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of $DC4_{PTMARK}$

66.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

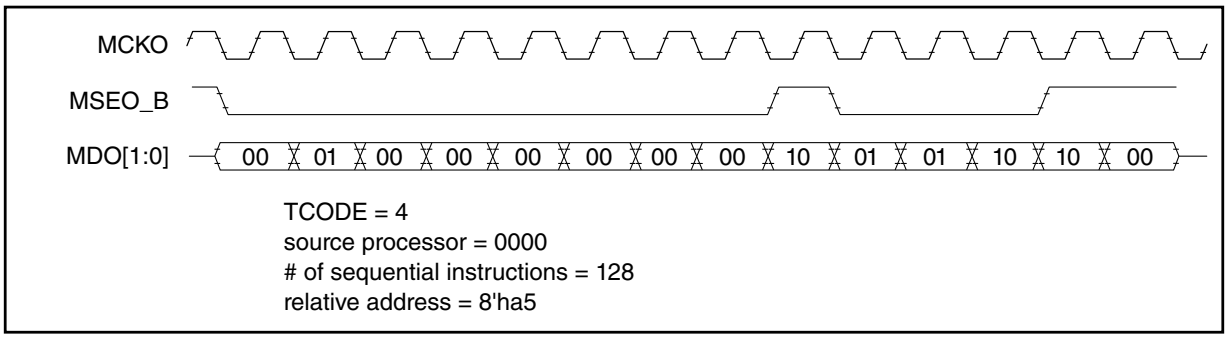


Figure 66-41. Program Trace - Indirect Branch message (Traditional)

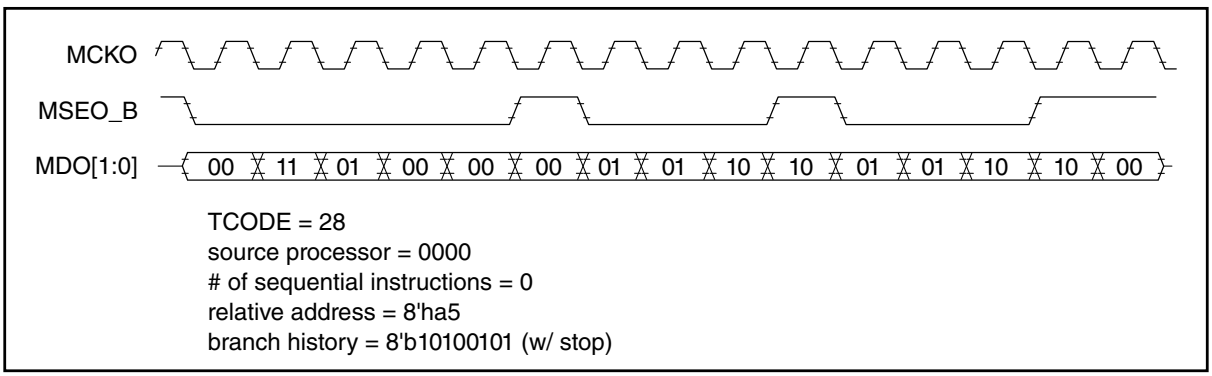


Figure 66-42. Program Trace - Indirect Branch message (history)

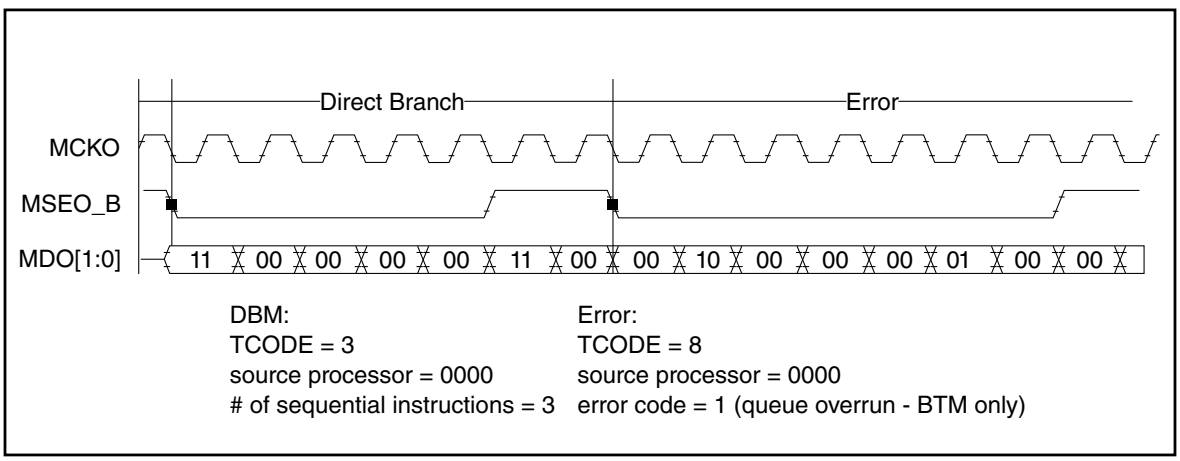


Figure 66-43. Program Trace - Direct Branch (traditional) & Error messages

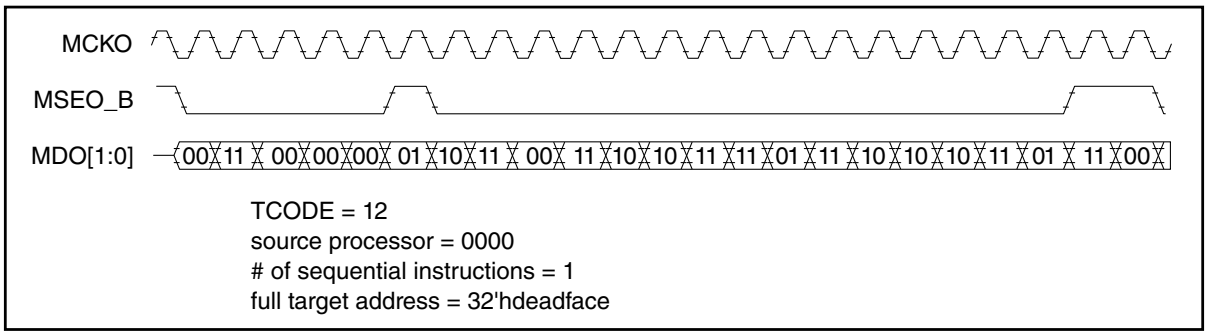


Figure 66-44. Program Trace - Indirect Branch w/ Sync. message

66.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

66.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note

Data Trace is only performed on the internal data buses. This allows for data visibility for Core (e200z210n3) processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of the following ways:

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).

66.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization and Error Messages.

66.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace Message. Data Write message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

Figure 66-45. Data Write message format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

66.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

Figure 66-46. Data Read fessage format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

Note

* Core (e200z210n3)-based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** =

0b000). In these cases, the Nexus 3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.

For Core (e200z210n3)-based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.

The debug/development tool will need to distinguish the two cases based on the family of processor.

66.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync. message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see Table 66-36):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message) or suppression, provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.
- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

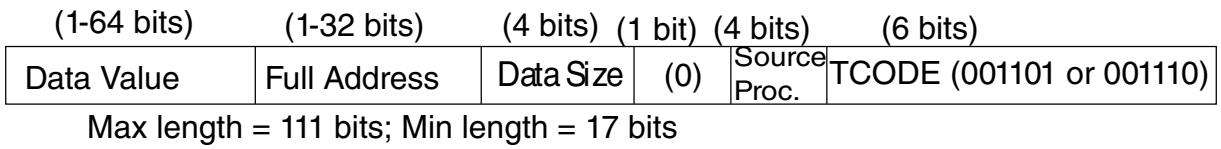


Figure 66-47. Data Write/Read with Sync. message format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

Exception conditions that result in Data Trace Synchronization are summarized in Table 66-36.

Table 66-36. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

66.13.3 DTM Operation

66.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

66.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

66.13.3.3 Data Trace filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and $DC4_{DTMARK}$.

66.13.3.4 Bus Cycle special cases

Table 66-37. Bus Cycle Special Cases

Special Case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{TEA}$), [1]	Data Trace Message discarded
Bus cycle completed without error[1]	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC_{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) - both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)

Table continues on the next page...

Table 66-37. Bus Cycle Special Cases (continued)

Special Case	Action
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note

For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (i.e. word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the doubleword at address 0. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Core (e200z210n3) Core Debug Support chapter.)

Note

A store to the store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

66.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)

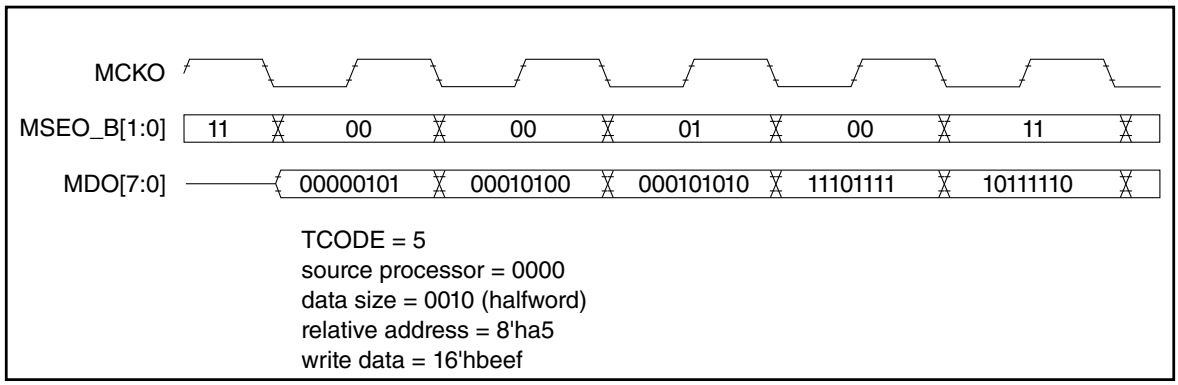


Figure 66-48. Data Trace - Data Write Message

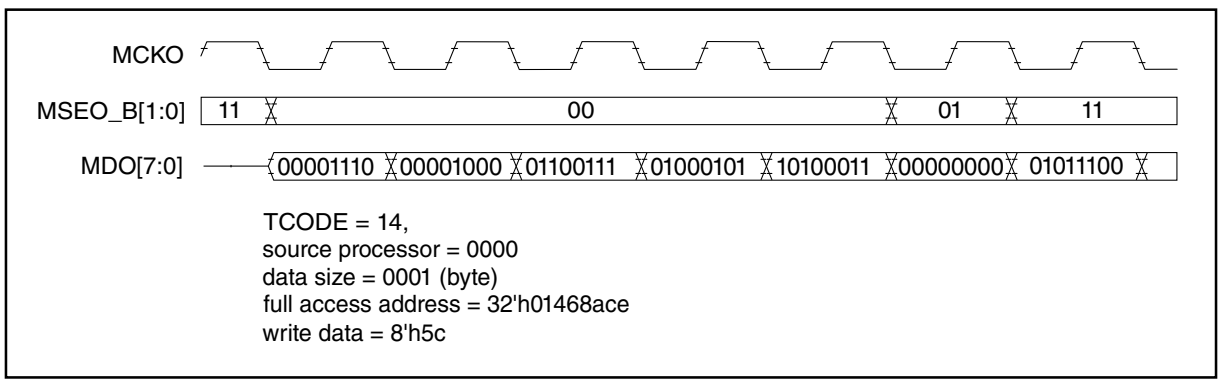


Figure 66-49. Data Trace — Data Read w/ Sync message

66.14 Data Acquisition messaging

This section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

66.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 8-bit value specifying control or attribute information for the data included in the Data Acquisition message. DQTAG is sampled from $DEVENT_{DQTAG}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

66.14.2 Data Acquisition Data field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

66.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the "External Debug Resource Allocation Control Register (EDBRAC0)" section in the Core (e200z210n3) Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.

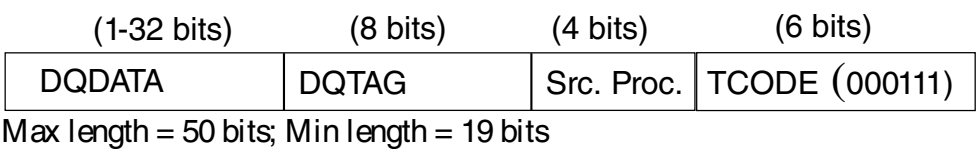


Figure 66-50. Data Acquisition message format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

66.15 Watchpoint Trace messaging

Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Core (e200z210n3) Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the $WMSK_{WEM}$ field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See Table 66-40 for details on **evto_b**.

Watchpoint information is messaged out in the following format:

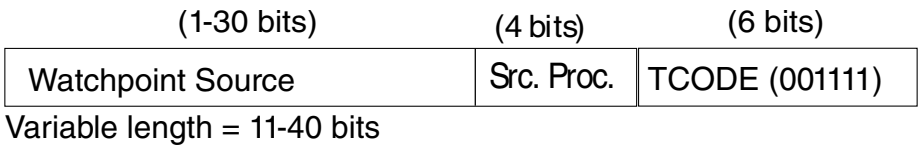


Figure 66-51. Watchpoint message format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

The Watchpoint Source message field will contain a '1' for each asserted watchpoint. Leading zeros are truncated.

Table 66-38. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
000000000000000000000000000000 - No Watchpoints enabled for Watchpoint Trace Messaging	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 - Watchpoint #0 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X - Watchpoint #1 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX - Watchpoint #2 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX - Watchpoint #3 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX - Watchpoint #4 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX - Watchpoint #5 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX - Watchpoint #6 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX - Watchpoint #7 enabled for WTM	
XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX - Watchpoint #8 enabled for WTM	
XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX - Watchpoint #9 enabled for WTM	
XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX - Watchpoint #10 enabled for WTM	
XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX - Watchpoint #11 enabled for WTM	
XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX - Watchpoint #12 enabled for WTM	
XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX - Watchpoint #13 enabled for WTM	

Table 66-38. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX	Watchpoint #14 enabled for WTM
XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX	Watchpoint #15 enabled for WTM
XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX	Watchpoint #16 enabled for WTM
XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX	Watchpoint #17 enabled for WTM
XXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX	Watchpoint #18 enabled for WTM
XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX	Watchpoint #19 enabled for WTM
XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX	Watchpoint #20 enabled for WTM
XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX	Watchpoint #21 enabled for WTM
XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #22 enabled for WTM
XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #23 enabled for WTM
XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #24 enabled for WTM
XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #25 enabled for WTM
XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #26 enabled for WTM
XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #27 enabled for WTM
X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #28 enabled for WTM
1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	Watchpoint #29 enabled for WTM

66.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

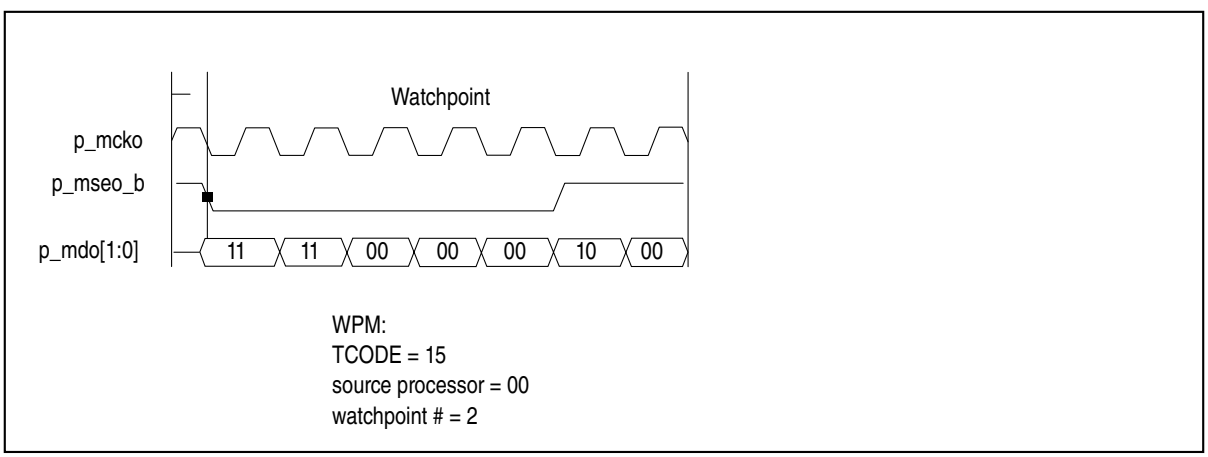


Figure 66-52. Watchpoint Message & Watchpoint Error message

66.16 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6'b000011 accordingly. The $RWCS_{ATTR}$ field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note

Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

66.16.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Address → 32h'xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status (RWCS) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:

- Access Control (AC) → 1b'1 (to indicate start access)
- Map Select (MAP) → 3b'000 (primary memory map)
- Access Priority (PR) → 2b'11 (highest priority)
- Read/Write (RW) → 1b'1 (write access)
- Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
- Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note

Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data (RWD) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Data → 32h'xxxxxxxx (write data)
4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD register to the memory mapped address in the Read/Write Access Address (RWA) register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed, Nexus clears the AC and DV bits in the RWCS register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (see Table 66-40 for details) and clears the ERR bit in the RWCS register. Otherwise, if the access completes with an error, the **nex_err_b** pin will be asserted and the ERR bit will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once the DV bit has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

Note

Only the **nex_ahb_start**,**nex_err_b**, and **nex_rdy_b** pins, as well as the AC, DV, and ERR bits within the RWCS, provide Read/Write Access status to the external development tool.

66.16.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in Single write access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The Nexus block then arbitrates for the AHB system bus and transfers the first data value from the RWD register to the memory-mapped address in the Read/Write Access Address (RWA) register. The nex_ahb_start output is asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, and the **nex_rdy_b** pin is asserted. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS register. Otherwise, the ERR bit is set and the DV bit is cleared to indicate an error has occurred, the **nex_err_b** pin is asserted, the block transfer is aborted, and the AC bit in the RWCS register is cleared. Clearing the DV bit indicates that the device is ready for the next access in the block transfer, or that an error has occurred.
3. If the AC bit has not been cleared due to an error, repeat Step 3 in Single write access until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS register is cleared to indicate the end of the block write access.

Note

The actual RWA register value as well as the CNT field within the RWCS register are not changed when executing a block write access. The original values can be read by the external development tool at any time.

66.16.3 Single read access

1. Initialize the Read/Write Access Address (RWA)register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Read Address → 32h'xxxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status (RWCS) register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)

- Access Priority (PR) → 2b'11 (highest priority)
- Read/Write (RW) → 1b'0 (read access)
- Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
- Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note

An Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS register and asserts the **nex_rdy_b** pin (see Table 66-40 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set, and the DV bit will be cleared to indicate an error has occurred. The **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit is cleared in either case.
4. The data can then be read from the Read/Write Access Data register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.

Note

Only the **nex_ahb_start**, **nex_err_b**, and **nex_rdy_b** pins as well as the AC, DV, and ERR bits within the RWCS register provide Read/Write Access status to the external development tool.

66.16.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in Single read access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access

has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS register and asserts the **nex_rdy_b** pin (see Table 66-40 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.

3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the RWCS register, and asserts the **nex_rdy_b** pin (see Table 66-40 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.
4. The data can then be read from the RWD register through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in Single read access until the CNT value is zero (0). When this occurs, the AC bit within the RWCS register is cleared to indicate the end of the block read access.

Note

The data values must be shifted out 32 bits at a time LSB first (i.e. doubleword read = two word reads from the RWD).

Note

The actual RWA value as well as the CNT field within the RWCS register are not changed when executing a block read access. The original values can be read by the external development tool at any time.

66.16.5 Error handling

The Nexus 3 module handles various error conditions as follows:

66.16.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (AC bit is cleared)
2. The ERR bit in the RWCS Register is set
3. The Error Message is sent (TCODE = 8) indicating Read/Write error

66.16.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
    a. If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
    b. If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

66.16.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

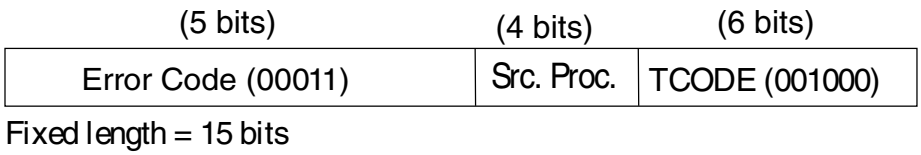


Figure 66-53. Error message format1

66.17 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

66.17.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_rdy_b** pin, (1) **nex_err_b** pin, (1) **evto_b** pin, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in Table 66-40. The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see Nexus 3 Register Access via JTAG/OnCE for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the "JTAG/OnCE Pins" section of the Core (e200z210n3) Core Debug Support chapter.

Table 66-39. JTAG Pins for Nexus 3

JTAG Pins	Input/ Output	Description of JTAG Pins (included in Nexus 1)
j_tdo	O	The Test Data Output (**j_tdo**) pin is the serial output for test instructions and data. **j_tdo** is three-stateable and is actively driven in the "Shift-IR" and "Shift-DR" controller states. **j_tdo** changes on the falling edge of **j_tclk**.
j_tdi	I	The Test Data Input (**j_tdi**) pin receives serial test instruction and data. TDI is sampled on the rising edge of **j_tclk**.
j_tms	I	The Test Mode Select (**j_tms**) input pin is used to sequence the OnCE controller state machine. **j_tms** is sampled on the rising edge of **j_tclk**.
j_tclk	I	The Test Clock (**j_tclk**) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (**j_trst_b**) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in Table 66-40.

Table 66-40. Nexus 3 Auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (**nex_mcko**) is a free running output clock to development tools for timing of **nex_mdo[n:0]** & **nex_mseo_b[1:0]** pin functions. **nex_mcko** is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (**nex_mdo[n:0]**) are output pins used for OTM, BTM, and DTM. External latching of **nex_mdo[n:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**).
nex_mseo_b[1:0]	O	Message Start/End Out (**nex_mseo_b[1:0]**) are output pins that indicate when a message on the **nex_mdo[n:0]** pins has started, when a variable length packet has ended, and when the message has ended. External latching of **nex_mseo_b[1:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**). One or two pin MSEO functionality is determined at integration time per SOC implementation.
nex_ahb_start	O	AHB Start (**nex_ahb_start**) is an output pin used to indicate to the external tool or SoC that the Nexus block is requesting the next Read/Write Access on the system bus. If Nexus is enabled, this signal is asserted upon an acknowledged request to start an AHB system bus transfer (Nexus read or write) and is pulsed asserted for one **nex_clk** clock period, corresponding to the first clock of the address phase of the transfer. Upon exit from system reset or if Nexus is disabled, **nex_ahb_start** remains de-asserted.
nex_rdy_b	O	Ready (**nex_rdy_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access. If Nexus is enabled, this signal is asserted upon successful (without error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the "Capture_DR" state. Upon exit from system reset or if Nexus is disabled, **nex_rdy_b** remains deasserted.
nex_err_b	O	Error (**nex_err_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access and an error has occurred on the previous access. If Nexus is enabled, this signal is asserted upon unsuccessful (with error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the "Capture_DR" state. Upon exit from system reset or if Nexus is disabled, **nex_err_b** remains deasserted.
evto_b	O	Event Out (**evto_b**) is an output that, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. **evto_b** is held asserted for one (1) cycle of **nex_mcko**: 1. One (or more) watchpoints has occurred (from Nexus1) & EOC = 2'b00 2. Debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2'b01
nex_evti_b	I	Event In (**nex_evti_b**) is an input that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1. Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2'b00). 2. Debug request to Nexus1 module (provided EIC = 2'b01 and this feature is implemented).
nex_ext_src_id[0:3]	I	**nex_ext_src_id[0:3]** is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time.

Table continues on the next page...

Table 66-40. Nexus 3 Auxiliary pins (continued)

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_sfwcntl_en]	I	**nex_sfwcntl_en** is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to Auxiliary port arbitration for detail on Nexus port arbitration.

Table 66-41. Nexus port arbitration signals

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (**nex_aux_req[1:0]**) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See Table 66-43.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.
nar_aux_grant	I	Nexus Auxiliary Grant **(nar_aux_grant)** is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (**ext_multi_nex_sel**) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

66.17.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in Table 66-42. Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

Single pin MSEO is not supported on the .

Table 66-42. MSEO pin(s) protocol

nex_mseo_b function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1's)	00 (or 01)-11-(more 1's)
End of variable length packet	0-1-0	00-01
Message transmission	0's	00's
Idle (no message)	1's	11's

Figure 66-54 illustrates the state diagram for single pin MSEO transfers.

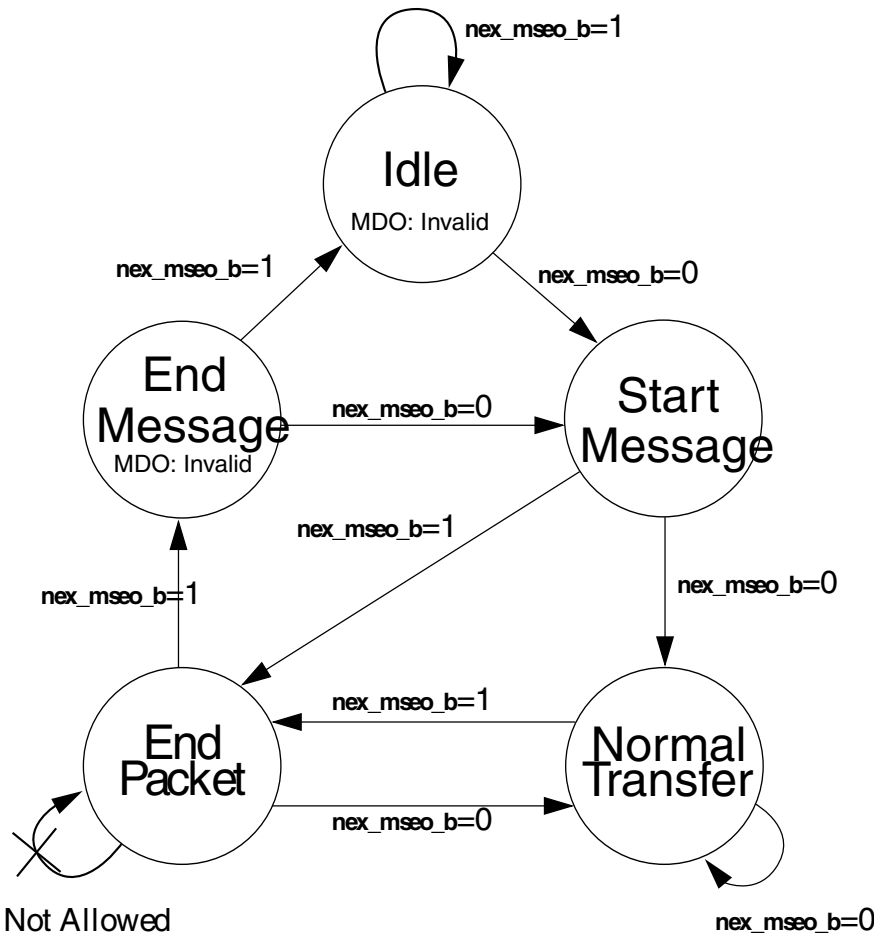


Figure 66-54. Single pin MSEO transfers

Note that the "End Message" state does not contain valid data on the **nex_mdo[n:0]** pins. Also, It is not possible to have two consecutive "End Packet" messages. This implies the minimum packet size for a variable length packet is 2x the number of **nex_mdo[n:0]** pins. This ensures that a false end of message state is not entered by emitting two consecutive '1's on the **nex_mseo_b** pin before the actual end of message.

Figure 66-55 illustrates the state diagram for dual pin MSEO transfers.

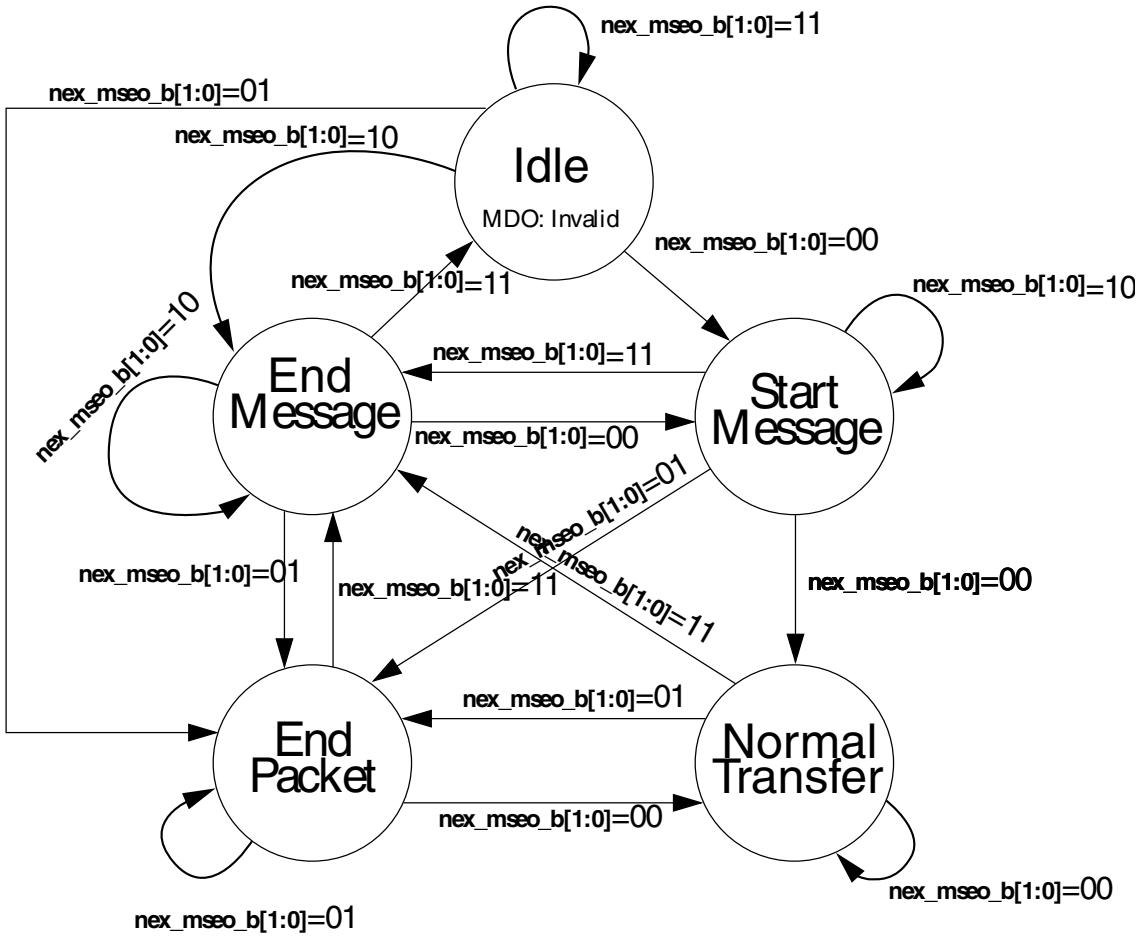


Figure 66-55. Dual pin MSEO transfers

The dual pin MSEO option is more robust that the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive "End Packet" states. This can be an advantage when small, variable sized packets are transferred.

Note

The "End Message" state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

66.18 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is impossible to know which bit is from the last packet and which bit is from the next packet.)
- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.

  For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

66.19 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level.The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in Table 66-43 below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in Pin protocol. The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the $\overline{\text{MSEO}}$ state machine reaches the "End Message" state.

Table 66-43. MDO Request Encodings

Request level	MDO request encoding (nex_aux_req[1:0])	Condition of queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

66.20 Examples

The following are examples of Program Trace and Data Trace Messages.

Table 66-44 illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. Table 66-45 illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 66-44. Indirect Branch Message Example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 66-45. Indirect branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I4	I3	I2	I1	I0	M A	S3	S2	0	1	End Packet

Table continues on the next page...

Table 66-45. Indirect branch message example (8 MDO / 2 MSEO) (continued)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
						P					
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 66-46 and Table 66-47 illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 66-46. Direct branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 66-47. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 66-48 illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and Table 66-49 illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 66-48. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 66-49. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

66.21 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, please refer to the Data Sheet.

66.22 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

66.22.1 JTAG sequence for accessing internal Nexus registers

Table 66-50. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE -> SELECT-DR_SCAN
2	0	SELECT-DR_SCAN -> CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR -> SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR -> EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR -> UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR -> SELECT-DR_SCAN
8	0	SELECT-DR_SCAN -> CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR -> SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR -> EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR -> UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR -> RUN-TEST/IDLE (transfer complete - Nexus controller to Reg. Select state)

66.22.2 JTAG sequence for read access of memory-mapped resources

Table 66-51. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

66.22.3 JTAG sequence for write access of memory-mapped resources

Table 66-52. Accessing memory-mapped resources (writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value - data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address - data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b**pin
8	—	If CNT > 0, go back to Step #5

Chapter 67
Z4 Nexus 3 Module

67.1 Introduction

The e200z4204n3 Nexus3 module provides real-time development capabilities for corresponding core processors in compliance with the IEEE-ISTO 5001 standard. This module provides development support capabilities without requiring the use of address and data pins for internal visibility.

A portion of the pin interface (the JTAG port) is also shared with the OnCE / Nexus 1 unit. The IEEE-ISTO 5001 standard defines an extensible auxiliary port which is used in conjunction with the JTAG port in these processors.

67.1.1 General Description

This chapter defines the auxiliary pin functions, transfer protocols and standard development features of a Class 3 device in compliance with the IEEE-ISTO 5001 standard. The development features supported are Program Trace, Data Trace, Watchpoint Messaging, Ownership Trace, Data Acquisition Messaging, and Read/Write Access via the JTAG interface. The Nexus 3 module also supports two Class 4 features: Watchpoint Triggering, and Processor Overrun Control.

67.1.2 Terms and Definitions

The following table contains a set of terms and definitions associated with the Nexus 3 module.

Table 67-1. Terms and Definitions

Term	Description
IEEE-ISTO 5001	Consortium & standard for real-time embedded system design. World wide Web documentation at http://www.ieee-isto.org/Nexus5001
Auxiliary Port	Refers to Nexus auxiliary port. Used as auxiliary port to the IEEE 1149.1 JTAG interface.
Branch Trace Messaging (BTM)	Visibility of addresses for taken branches and exceptions, and the number of sequential instructions executed between each taken branch.
Data Read Message (DRM)	External visibility of data reads to memory-mapped resources.
Data Write Message (DWM)	External visibility of data writes to memory-mapped resources.
Data Trace Messaging (DTM)	External visibility of how data flows through the embedded system. This may include DRM and/or DWM.
Data Acquisition Messaging (DQM)	Data Acquisition Messaging (DQM) allows code to be instrumented to export customized information to the Nexus Auxiliary Output Port.
JTAG Compliant	Device complying to IEEE 1149.1 JTAG standard
JTAG IR & DR Sequence	JTAG Instruction Register (IR) scan to load an opcode value for selecting a development register. The JTAG IR corresponds to the OnCE command register (OCMD). The selected development register is then accessed via a JTAG Data Register (DR) scan.
Nexus1	The (OnCE) debug module. This module integrated with each processor provides all static (core halted) debug functionality. This module is compliant with Class1 of the IEEE-ISTO 5001 standard.
Ownership Trace Message (OTM)	Visibility of process/function that is currently executing.
Public Messages	Messages on the auxiliary pins for accomplishing common visibility and controllability requirements
SoC	"System-on-a-Chip". SoC signifies all of the modules on a single die. This generally includes one or more processors with associated peripherals, interfaces & memory modules.
Standard	The phrase "according to the standard" is used to indicate according to the IEEE-ISTO 5001 standard.
Transfer Code (TCODE)	Message header that identifies the number and/or size of packets to be transferred, and how to interpret each of the packets.
Watchpoint	A Data or Instruction Breakpoint or other debug event that does not cause the processor to halt. Instead, a pin is used to signal that the condition occurred. A Watchpoint Message may also be generated.

67.1.3 Feature List

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

- Program Trace via Branch Trace Messaging (BTM). Branch trace messaging displays program flow discontinuities (direct and indirect branches, exceptions, etc.), allowing the development tool to interpolate what transpires between the discontinuities. Thus static code may be traced.
- Data Trace via Data Write Messaging (DWM) and Data Read Messaging (DRM). This provides the capability for the development tool to trace reads and/or writes to selected internal memory resources.
- Ownership Trace via Ownership Trace Messaging (OTM). OTM facilitates ownership trace by providing visibility of which process ID or operating system task is activated. An Ownership Trace Message is transmitted when a new process/task is activated, allowing the development tool to trace ownership flow.
- Run-time access to embedded processor memory map via the JTAG port. This allows for enhanced download/upload capabilities.
- Watchpoint Messaging via the auxiliary pins
- Watchpoint Trigger enable of Program and/or Data Trace Messaging
- Auxiliary interface for higher data input
  - Configurable (min/max) Message Data Out pins (**nex_mdo[n:0]**)
  - One or two Message Start/End Out pins (**nex_mseo_b[1:0]**)
  - One Read/Write Error pin (**nex_err_b**) pin
  - One Watchpoint Event output pin (**nex_evto_b**)
  - One Event In pin (**nex_evti_b**)
  - One MCKO (Message Clock Out) pin
- Registers for Program Trace, Data Trace, Ownership Trace and Watchpoint Trigger.
- All features controllable and configurable via the JTAG port
- Conditional software control of the module via device signaling input (**nex_sfwcntl_en**)

Note

For multi-Nexus implementations, the configuration of the Message Data Out pins is controlled by the Port Control Register (at the device level). For single Nexus

implementations (not normally implemented on an SoC), this configuration is controlled by Development Control Register 1 (DC1) within the Nexus 3 module.

In either implementation, Full Port Mode (FPM - maximum number of MDO pins) or Reduced Port Mode (RPM - minimum number of MDO pins) are supported. This setting should not be changed while the system is running.

Note

The configuration of the Message Start/End Out pins (1 or 2) is determined at the device integration level. This option will be hard-wired based on device bandwidth requirements.

67.1.4 Functional Block Diagram

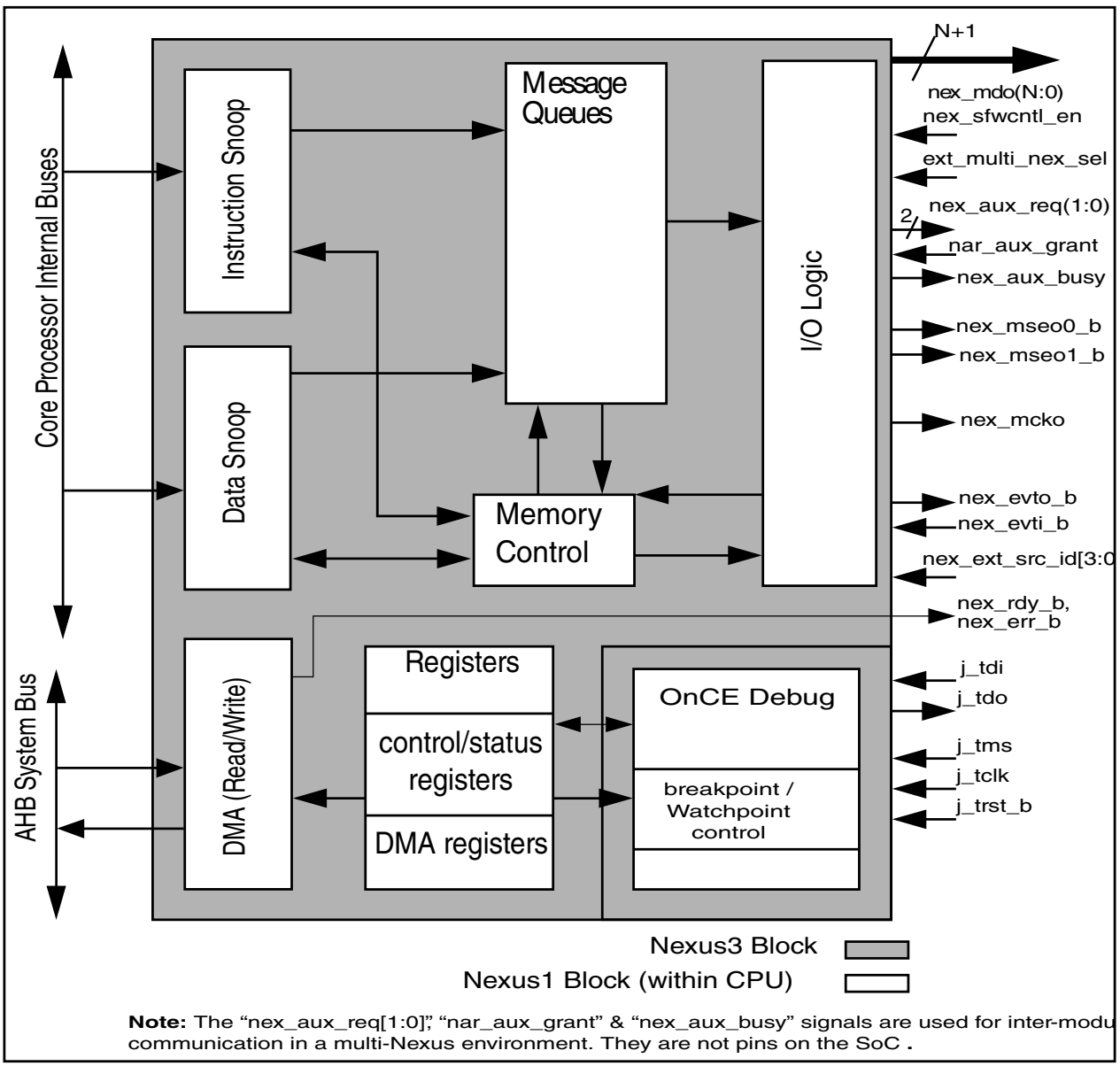


Figure 67-1. Nexus 3 Functional Block Diagram

67.2 Enabling Nexus 3 Operation

The Nexus module is enabled by loading a single instruction (*NEXUS3-ACCESS*) into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus3 module, the OCMD value is 0b0001111100. The Nexus 3 module may alternately be enabled if the nex_evti_b input signal is asserted at the time that j_trst_b is initially negated, or is asserted during the Test-Logic-Reset TAP state. Once enabled, the module will be ready to accept control input via the JTAG/OnCE pins.

Enabling the Nexus 3 module automatically enables the generation of Debug Status Messages.

The Nexus 3 module is disabled when the JTAG state machine initially reaches the Test-Logic-Reset state from any other state. This state can be reached by the assertion of the **j_trst_b** pin or by cycling through the state machine using the **j_tms** pin. The Nexus module will also be disabled if a Power-on-Reset (POR) event occurs. If the Nexus 3 module is disabled, no trace output will be provided, and the module will disable (drive inactive) auxiliary port output pins (**nex_mdo[n:0]**, **nex_mseo[1:0]**, **nex_mcko**). Nexus registers will not be available for reads or writes.

In order to support software control of the Nexus 3 module when no external development tool is present, the Nexus 3 module is not forced to be disabled when the JTAG state remains in the Test-Logic-Reset state. Software is allowed to control the Nexus 3 module when the input signal **nex_sfwcntl_en** is asserted by the SoC. This signal is intended to provide a mechanism for allowing software to use the Nexus 3 module to fill on-chip trace buffers or other visibility mechanisms. It is up to the SoC to determine whether a top-level Nexus 3 controller has been enabled by a hardware debugger, or whether appropriate security mechanisms have granted the capability for software to use these resources, and to drive the appropriate value to the **nex_sfwcntl_en** input. Software can enable the module by a write to any of the module's DCRs when **nex_sfwcntl_en** is asserted.

Reset of the Nexus 3 module is accomplished by a transition on **j_trst_b**, on initial entry into the Test_Logic_Reset state from another state, or if a Power-on-Reset (POR) event occurs. The module is not reset by the CPU's **p_reset_b** signal, even when software has control of the module.

67.2.1 Interaction with Low Power Modes

The Nexus 3 module will continue to operate in the Waiting and Halted states, as long as **nex_clk** remains active. In the Stopped state, **nex_clk** is gated off internally to the Nexus 3 logic, therefore watchpoint or hardware triggering recognition, message generation, and Nexus 3 read-write access to memory is suspended. The Nexus 3 logic will wait to enter the Stopped state until the message FIFOs are empty and any in-progress Nexus R/W transfer has completed. If a block transfer has been requested, the remainder of the block transfer is not completed, and the RWCS AC bit is cleared, and the ERR bit is set. Once the Stopped state has been entered, no further messages are queued and no triggering conditions are monitored. Also, Nexus R/W accesses are no longer available. Upon exiting the Stopped state, normal functions will resume assuming **nex_clk** is active.

67.3 TCODEs supported

The Nexus 3 pins allow for flexible transfer operations via Public Messages. A TCODE defines the transfer format, the number and/or size of the packets to be transferred, and the purpose of each packet. The IEEE-ISTO 5001 2003 standard defines a set of public messages and allocates additional TCODEs for vendor-specific features outside the scope of the public messages. The Nexus 3 block supports the TCODEs shown in Table 67-2.

Table 67-2. Supported TCODEs

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Debug Status	6	6	TCODE	fixed	TCODE number = 0
	4	4	SRC	fixed	source processor identifier
	8	8	STATUS	fixed	Development Status Register (DS[31:24])
Ownership Trace Message	6	6	TCODE	fixed	TCODE number = 2
	4	4	SRC	fixed	source processor identifier
	1	32	PROCESS	variable	Task/Process ID tag
Program Trace - Direct Branch Message	6	6	TCODE	fixed	TCODE number = 3
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
Program Trace - Indirect Branch Message	6	6	TCODE	fixed	TCODE number = 4
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions
Data Trace - Data Write Message	6	6	TCODE	fixed	TCODE number = 5
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 67-7)
	1	32	U-ADDR	variable	unique portion of the data write address
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace - Data Read Message	6	6	TCODE	fixed	TCODE number = 6
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 67-7)

Table continues on the next page...

Table 67-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	32	U-ADDR	variable	unique portion of the data read address
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Data Acquisition Message	6	6	TCODE	fixed	TCODE number = 7
	4	4	SRC	fixed	source processor identifier
	8	8	DQTAG	fixed	identification tag taken from $DEVENT_{DQTAG}$ register field
	1	32	DQDATA	variable	exported data taken from DDAM register
Error Message	6	6	TCODE	fixed	TCODE number = 8
	4	4	SRC	fixed	source processor identifier
	4	4	ETYPE	fixed	error type
	8	8	ECODE	fixed	error code
Program Trace - Synchronization Message	6	6	TCODE	fixed	TCODE number = 9
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow. Cleared for most sync conditions.
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
Program Trace - Direct Branch Message w/ Sync	6	6	TCODE	fixed	TCODE number = 11
	4	4	SRC	fixed	source processor identifier
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Program Trace - Indirect Branch Message w/Sync	6	6	TCODE	fixed	TCODE number = 12
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	ICNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zeros truncated)
Data Trace - Data Write Message w/ Sync	6	6	TCODE	fixed	TCODE number = 13
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 67-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data write value(s) (see Data Trace section for details)
Data Trace -	6	6	TCODE	fixed	TCODE number = 14
	4	4	SRC	fixed	source processor identifier

Table continues on the next page...

Table 67-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
Data Read Message w/ Sync	1	1	MAP	fixed	(always set to 0)
	4	4	DSZ	fixed	data size (Refer to Table 67-7)
	1	32	F-ADDR	variable	full access address (leading zeros truncated)
	1	64	DATA	variable	data read value(s) (see Data Trace section for details)
Watchpoint Message	6	6	TCODE	fixed	TCODE number = 15
	4	4	SRC	fixed	source processor identifier
	1	32	WPHIT	variable	Field indicating watchpoint source(s) (leading zeros truncated)
Resource Full Message	6	6	TCODE	fixed	TCODE number = 27
	4	4	SRC	fixed	source processor identifier
	4	4	RCODE	fixed	resource code (Refer to Table 67-5) - indicates which resource is the cause of this message
	1	32	RDATA	variable	branch / predicate instruction history (See Section Resource Full Messages.)
Program Trace - Indirect Branch History Message	6	6	TCODE	fixed	TCODE number = 28 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	U-ADDR	variable	unique part of target address for taken branches/ exceptions
	1	32	HIST	variable	branch / predicate instruction history (see Branch Trace Messaging types)
Program Trace - Indirect Branch History Message w/ Sync	6	6	TCODE	fixed	TCODE number = 29 (see Note below)
	4	4	SRC	fixed	source processor identifier
	1	1	MAP	fixed	(always set to 0)
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	F-ADDR	variable	full target address (leading zero (0) truncated)
	1	32	HIST	variable	branch / predicate instruction history (See Branch Trace Messaging types.)
Program Trace - Program Correlation Message	6	6	TCODE	fixed	TCODE number = 33
	4	4	SRC	fixed	source processor identifier
	4	4	EVCODE	fixed	event correlated w/ program flow (Refer to Table 67-6)
	2	2	CDF	fixed	# fields of information in CDATA. 00 - reserved, 01 - one field (CDATA1) (reserved), 10 - two fields (CDATA1 + CDATA2), 11 - three fields (reserved)

Table continues on the next page...

Table 67-2. Supported TCODEs (continued)

Message Name	Min Field Size (bits)	Max Field Size (bits)	Field Name	Field Type	Field Description
	1	8	I-CNT	variable	# sequential instructions completed since last predicate instruction, transmitted instruction count, or taken change of flow
	1	32	CDATA1	variable	correlation data field 1 - [branch / predicate instruction history] (See Program Correlation Messages.)
	0	32	CDATA2	variable	correlation data field 2- PID info (See Program Correlation Messages.)

Note

Program Trace can be implemented using either Branch History/Predicate Instruction Messages, or traditional Direct/ Indirect Branch Messages. The user can select between the two types of Program Trace. The advantages for each are discussed in Branch Trace Messaging types. If the Branch History method is selected, the shaded TCODES above will not be messaged out.

Table 67-3 shows the error code encodings used when reporting an error via the Nexus 3 Error Message.

Table 67-3. Error Code (ECODE) Encoding (TCODE = 8)

Error Code	Description
xxxxxxx1	Watchpoint Trace Message(s) Lost
xxxxxx1x	Data Trace Message(s) Lost
xxxxx1xx	Program Trace Message(s) Lost
xxxx1xxx	Ownership Trace Message(s) Lost
xxx1xxxx	Status Message(s) Lost (Debug Status messages, etc.)
xx1xxxxx	Data Acquisition Message(s) Lost
x1xxxxxx	Reserved
1xxxxxxx	Reserved

Table 67-4 shows the error type encodings used when reporting an error via the Nexus 3 Error Message.

Table 67-4. Error Type (ETYPE) Encoding (TCODE = 8)

Error Type	Description
0000	Message Queue Overrun caused one or more messages to be lost

Table continues on the next page...

Table 67-4. Error Type (ETYPE) Encoding (TCODE = 8) (continued)

Error Type	Description
0001	Contention with higher priority messages caused one or more messages to be lost
0010	Reserved
0011	Reserved
0100	Reserved
0101	Invalid access opcode (Nexus Register unimplemented)
0110 - 1111	Reserved

Table 67-5 shows the encodings used for resource codes for certain messages.

Table 67-5. Resource Code (RCODE) values (TCODE = 27)

Resource Code	Description
0000	Program Trace Instruction counter reached 255 and was reset.
0001	Program Trace, Branch / Predicate Instruction History full. This type of packet is terminated by a stop bit set to 1 after the last history bit.

Table 67-6 shows the event code encodings used for certain messages.

Table 67-6. Event Code (EVCODE) Encoding (TCODE = 33)

Event Code	Description
0000	Entry into Debug Mode
0001	Entry into Low Power Mode (CPU only)
0010-0011	Reserved for future functionality
0100	Disabling Program Trace
0101	Process ID value is established in PID0/NPIDR via **mtspr PID0/NPIDR**
0110-1000	Reserved for future functionality
1001	Begin masking of program trace messages due to MSR_{PMM}=0 and $DC4_{PTMARK}$=1
1010	Branch and link occurrence (direct branch function call)
1011-1111	Reserved for future functionality

Table 67-7 shows the data trace size encodings used for certain messages.

Table 67-7. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14)

DTM Size Encoding	Transfer Size
0000	0 - no data
0001	Byte
0010	Halfword (2 bytes)
0011	Three bytes
0100	Word (4 bytes)

Table continues on the next page...

Table 67-7. Data Trace Size (DSZ) Encodings (TCODE = 5,6,13,14) (continued)

DTM Size Encoding	Transfer Size
0101	Five bytes
0110	Six bytes
0111	Seven bytes
1000	Doubleword (8 bytes)
1001-1111	Reserved

67.4 Nexus 3 Programmer's Model

This section describes the Nexus 3 programmers model. Nexus 3 registers are accessed using the JTAG/OnCE port in compliance with IEEE 1149.1, or by software via DCRs. See Nexus 3 Register Access via JTAG/OnCE and Nexus 3 Register Access via Software for details on Nexus 3 register access.

Note

Nexus 3 registers and output signals are numbered using bit 0 as the least significant bit. This bit ordering is consistent with the ordering defined by the IEEE-ISTO 5001 standard.

The following table details the register map for the Nexus 3 module.

Table 67-8. Nexus 3 Register Map

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #[1]
Client Select Control (CSC)[2]	0x1	R	0x02	-	
Port Configuration Register (PCR)[3]	PCR_INDEX[2]	R/W	-	-	
Development Control 1 (DC1)	0x2	R/W	0x04	0x05	368
Development Control 2 (DC2)	0x3	R/W	0x06	0x07	369
Development Control 3 (DC3)	0x4	R/W	0x08	0x09	370
Development Control 4 (DC4)	0x5	R/W	0x0A	0x0B	371
Reserved	0x6	R/W	0x18	0x19	
Read/Write Access Control/Status (RWCS)	0x7	R/W	0x0E	0x0F	-
Reserved	0x8	R/W	0x18	0x19	
Read/Write Access Address (RWA)	0x9	R/W	0x12	0x13	-
Read/Write Access Data (RWD)	0xA	R/W	0x14	0x15	-
Watchpoint Trigger (WT)	0xB	R/W	0x16	0x17	375
Reserved	0xC	R/W	0x18	0x19	

Table continues on the next page...

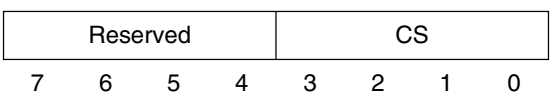
Table 67-8. Nexus 3 Register Map (continued)

Nexus Register	Nexus Access Opcode	Read/ Write	Read Address	Write Address	DCR #[1]
Data Trace Control (DTC)	0xD	R/W	0x1A	0x1B	376
Data Trace Start Address 1 (DTSA1)	0xE	R/W	0x1C	0x1D	377
Data Trace Start Address 2 (DTSA2)	0xF	R/W	0x1E	0x1F	378
Data Trace Start Address 3 (DTSA3)	0x10	R/W	0x20	0x21	379
Data Trace Start Address 4 (DTSA4)	0x11	R/W	0x22	0x23	380
Data Trace End Address 1 (DTEA1)	0x12	R/W	0x24	0x25	381
Data Trace End Address 2 (DTEA2)	0x13	R/W	0x26	0x27	382
Data Trace End Address 3 (DTEA3)	0x14	R/W	0x28	0x29	383
Data Trace End Address 4 (DTEA4)	0x15	R/W	0x2A	0x2B	408
Reserved	0x16 -> 0x2F	-	0x28->0x5E	0x29->5F	
Development Status (DS)	0x30	R	0x60	-	409
Reserved	0x31	R/W	0x62	0x63	
Overrun Control (OVCR)	0x32	R/W	0x64	0x65	410
Watchpoint Mask (WMSK)	0x33	R/W	0x66	0x67	411
Reserved	0x34	-	0x68	0x69	
Program Trace Start Trigger Control (PTSTC)	0x35	R/W	0x6A	0x6B	412
Program Trace End Trigger Control (PTETC)	0x36	R/W	0x6C	0x6D	413
Data Trace Start Trigger Control (DTSTC)	0x37	R/W	0x6E	0x6F	414
Data Trace End Trigger Control (DTETC)	0x38	R/W	0x70	0x71	415
Reserved	0x39 -> 0x3F	-	0x72->0x7E	0x73->7F	

1. Software access via the **mfdcr** and **mtdcr** instructions use these values for the DCR number. Software writes to these registers via **mtdcr** when **nex_sfwcntl_en** is negated are ignored.
2. The CSC and PCR registers are shown in this table as part of the Nexus programmer's model. They are only present at the top level SoC Nexus controller in a multi-Nexus implementation, not in the Nexus 3 module. The SoC's CSC Register is readable through Nexus, but the PCR is shown for reference only here.
3. The "PCR_INDEX" is a parameter determined by the SoC.

67.4.1 Client Select Control (CSC) register

The CSC Register determines which Nexus client is under development. This register is present at the top-level SOC Nexus 3 controller to select one of multiple on-chip Nexus 3 units.



Nexus Reg# - 0x1; Read-only; Reset - 0x0

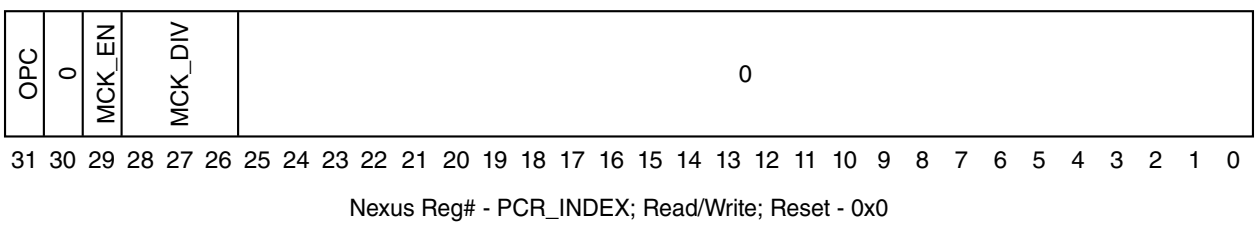
Figure 67-2. Client Select Control (CSC) register

Table 67-9. CSC field descriptions

Bits	Description
CSC[7:4]	Reserved for future Nexus Clients (read as 0)
CSC[3:0]	Client Select Control 0xx - Nexus client (SoC level)

67.4.2 Port Configuration Register (PCR) - reference only

The Port Configuration Register (PCR) controls the basic port functions for all Nexus modules in a multi-Nexus environment. This includes clock control and auxiliary port width. All bits in this register are writable only once after system reset.



Nexus Reg# - PCR_INDEX; Read/Write; Reset - 0x0

Figure 67-3. Port Configuration Register

Table 67-10. PCR field descriptions

Bit	Name	Description
31	OPC	Output Port Mode Control (SoC Level) 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined by SOC) 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined by SOC)
30	—	Reserved for future functionality
29	MCK_EN	MCKO Clock Enable (SoC Level) 0 **nex_mcko** is disabled 1 **nex_mcko** is enabled
28:26	MCK_DIV	MCKO Clock Divide Ratio (see note below) (SoC Level) 000 **nex_mcko** is 1x processor clock freq. 001 **nex_mcko** is 1/2x processor clock freq. 010 Reserved (default to 1/2x processor clock freq.) 011 **nex_mcko** is 1/4x processor clock freq. 100 Reserved (default to 1/2x processor clock freq.) 101 Reserved (default to 1/2x processor clock freq.) 110 Reserved (default to 1/2x processor clock freq.) 111 **nex_mcko** is 1/8x processor clock freq.
25:0	—	Reserved for future functionality

Note

The CSC and PCR Registers exist in a separate module at the SoC level in a multi-Nexus environment. If the core Nexus 3 module is the only Nexus module, these registers are not implemented and the Nexus 3 defined Development Control Register 1 (DC1) is used to control the SoC-level Nexus port functionality.

67.4.3 Nexus Development Control Register 1 (DC1)

Nexus Development Control Register 1 is used to control the basic development features of the Nexus 3 module. Development Control Register 1 is shown in the following figure and its fields are described in the following table.



Nexus Reg# - 0x2; DCR - 368; Read/Write; Reset - 0x0

Figure 67-4. Development Control Register 1

Table 67-11. DC1 field descriptions

Bits	Name	Description
31	OPC	Output Port Mode Control 0 Reduced Port Mode configuration (min# **nex_mdo[n:0]** pins defined 1 Full Port Mode configuration (max# **nex_mdo[n:0]** pins defined
30:29	MCK_DIV	MCKO Clock Divide Ratio (see note below) 00 **nex_mcko** is 1x processor clock freq. 01 **nex_mcko** is 1/2x processor clock freq. 10 **nex_mcko** is 1/4x processor clock freq. 11 **nex_mcko** is 1/8x processor clock freq.
28	—	Reserved for future functionality
27	PTM	Program Trace Method 0 Program Trace uses traditional Branch Messages 1 Program Trace uses Branch History Messages
26:25	—	Reserved for future functionality
24	OTS	Ownership Trace PID Select 0 PID0 data is transmitted within Ownership Trace Messages 1 Nexus PID Register (NPIDR) data is transmitted within Ownership Trace Messages
26:15	—	Reserved for future functionality

Table continues on the next page...

Table 67-11. DC1 field descriptions (continued)

Bits	Name	Description
14	POTD	Periodic Ownership Trace Disable 0 Periodic Ownership Trace message events are enabled 1 Periodic Ownership Trace message events are disabled
13:12	TSEN	Timestamp Enable - (not implemented, write to 00) 00 Timestamp is disabled
11:10	EOC	EVTO Control 00 **nex_evto_b** upon occurrence of Watchpoints (configured in DC2 and DC3) 01 **nex_evto_b** upon entry into Debug Mode 1x Reserved
9:8	EIC	EVTI Control 00 **nex_evti_b** is used for synchronization (Program Trace Sync msg is generated) 01 **nex_evti_b** is used for Debug request 10 **nex_evti_b** is disabled 1X Reserved
7:6	—	Reserved for future functionality
5:0	TM	Trace Mode This field may be updated by hardware in response to watchpoint triggering if not already enable for tracing by a previous direct write to DC1. Direct writes to this field to enable one or more trace types take precedence over hardware updates. Refer to Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers for more information on watchpoint triggering. 000000 All Trace Disabled XXXXX1 Ownership Trace enabled XXXX1X Data Trace enabled XXX1XX Program Trace enabled XX1XXX Watchpoint Trace enabled X1XXXX Reserved 1XXXXX Data Acquisition Trace enabled

Note

The Output Port Mode Control bit (OPC) and MCKO Clock Divide Ratio bits (MCK_DIV) MUST ONLY be modified during system reset or debug mode to insure correct output port and output clock functionality. It is also recommended that all other bits of the DC1 also only be modified in one of these two modes.

67.4.4 Nexus Development Control Registers 2 and 3 (DC2, DC3)

Nexus Development Control Registers 2 and 3 are used to control output signaling on the Nexus 3 module. A table of watchpoints can be found in the Z4 Core Core Debug Support chapter.

Development Control Register 2 is shown in Figure 67-5 and its fields are described in Table 67-12.

0	0	0	0	EWC

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x3; DCR - 369; Read/Write; Reset - 0x0

Figure 67-5. Development Control Register 2

Table 67-12. DC2 field descriptions

Bits	Name	Description
31:16	—	Reserved
15:0	EWC	EVTO Watchpoint Configuration[1] 0000000000000000 No Watchpoints #0-15 trigger **nex_evto_b** XXXXXXXXXXXXXXX1 Watchpoint #0 triggers **nex_evto_b** XXXXXXXXXXXXXX1X Watchpoint #1 triggers **nex_evto_b** XXXXXXXXXXXXX1XX Watchpoint #2 triggers **nex_evto_b** XXXXXXXXXXXX1XXX Watchpoint #3 triggers **nex_evto_b** XXXXXXXXXXX1XXXX Watchpoint #4 triggers **nex_evto_b** XXXXXXXXXX1XXXXX Watchpoint #5 triggers **nex_evto_b** XXXXXXXXX1XXXXXX Watchpoint #6 triggers **nex_evto_b** XXXXXXXX1XXXXXXX Watchpoint #7 triggers **nex_evto_b** XXXXXXX1XXXXXXXX Watchpoint #8 triggers **nex_evto_b** XXXXXX1XXXXXXXXX Watchpoint #9 triggers **nex_evto_b** XXXXX1XXXXXXXXXX Watchpoint #10 triggers **nex_evto_b** XXXX1XXXXXXXXXXX Watchpoint #11 triggers **nex_evto_b** XXX1XXXXXXXXXXXX Watchpoint #12 triggers **nex_evto_b** XX1XXXXXXXXXXXXX Watchpoint #13 triggers **nex_evto_b** X1XXXXXXXXXXXXXX Watchpoint #14 triggers **nex_evto_b** 1XXXXXXXXXXXXXXX Watchpoint #15 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

Development Control Register 3 is shown in Figure 67-6 and its fields are described in Table 67-13.

0	0	0	0	0	EWC

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x4; DCR - 370; Read/Write; Reset - 0x0

Figure 67-6. Development Control 3 (DCR3) register

Table 67-13. DC3 field descriptions

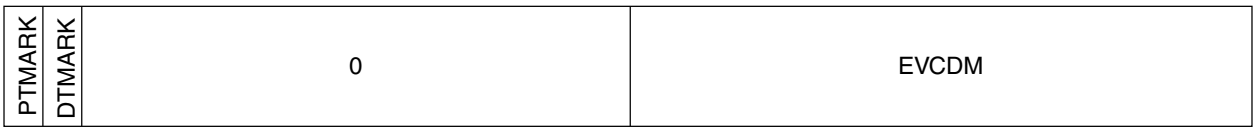
Bits	Name	Description
31:16	—	Reserved
15:14	—	Reserved for watchpoint expansion
13:0	EWC	EVTO Watchpoint Configuration[1] 00000000000000 No Watchpoints #16-#29 trigger **nex_evto_b** XXXXXXXXXXXXX1 Watchpoint #16 triggers **nex_evto_b** XXXXXXXXXXXX1X Watchpoint #17 triggers **nex_evto_b** XXXXXXXXXXX1XX Watchpoint #18 triggers **nex_evto_b** XXXXXXXXXX1XXX Watchpoint #19 triggers **nex_evto_b** XXXXXXXXX1XXXX Watchpoint #20 triggers **nex_evto_b** XXXXXXXX1XXXXX Watchpoint #21 triggers **nex_evto_b** XXXXXXX1XXXXXX Watchpoint #22 triggers **nex_evto_b** XXXXXX1XXXXXXX Watchpoint #23 triggers **nex_evto_b** XXXXX1XXXXXXXX Watchpoint #24 triggers **nex_evto_b** XXXX1XXXXXXXXX Watchpoint #25 triggers **nex_evto_b** XXX1XXXXXXXXXX Watchpoint #26 triggers **nex_evto_b** XX1XXXXXXXXXXX Watchpoint #27 triggers **nex_evto_b** X1XXXXXXXXXXXX Watchpoint #28 triggers **nex_evto_b** 1XXXXXXXXXXXXX Watchpoint #29 triggers **nex_evto_b**

1. EOC bits in DC1 must be programmed to trigger $\overline{\text{EVTO}}$ on Watchpoint occurrence for EWC bits to have any effect.

67.4.5 Nexus Development Control Register 4 (DC4)

Nexus Development Control Register 4 is used to control mark selection for Program and Data Trace Messaging, as well as masking of events that initiate Program Correlation Messages on the Nexus 3 module.

Development Control Register 4 is shown in Figure 67-7 and its fields are described in Table 67-14.



Nexus Reg# - 0x5; DCR - 371; Read/Write; Reset - 0x0

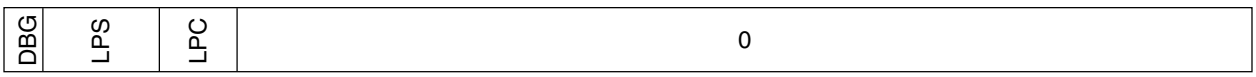
Figure 67-7. Development Control 4 (DC4) register

Table 67-14. DC4 field descriptions

Bits	Name	Description
31	PTMARK	Program Trace Mark 0 Ignore MSR_{PMM} for masking program trace messages 1 Mask program trace messages when MSR_{PMM}='0', unmask program trace messages when MSR_{PMM}='1'
30	DTMARK	DTMARK Data Trace Mark 0 Ignore MSR_{PMM} for masking data trace messages 1 Mask data trace messages when MSR_{PMM}='0', unmask data trace messages when MSR_{PMM}='1'
29:16	—	Reserved
15:0	EVCDM	Event Code (EVCODE) Mask For implemented EVCODES, please see Table 67-6. 0000000000000000 No EVCODEs masked for Program Correlation Messages XXXXXXXXXXXXXXX1 EVCODE #0 is masked for Program Correlation Messages XXXXXXXXXXXXXX1X EVCODE #1 is masked for Program Correlation Messages XXXXXXXXXXXXX1XX EVCODE #2 is masked for Program Correlation Messages XXXXXXXXXXXX1XXX EVCODE #3 is masked for Program Correlation Messages XXXXXXXXXXX1XXXX EVCODE #4 is masked for Program Correlation Messages XXXXXXXXXX1XXXXX EVCODE #5 is masked for Program Correlation Messages XXXXXXXXX1XXXXXX EVCODE #6 is masked for Program Correlation Messages XXXXXXXX1XXXXXXX EVCODE #7 is masked for Program Correlation Messages XXXXXXX1XXXXXXXX EVCODE #8 is masked for Program Correlation Messages XXXXXX1XXXXXXXXX EVCODE #9 is masked for Program Correlation Messages XXXXX1XXXXXXXXXX EVCODE #10 is masked for Program Correlation Messages XXXX1XXXXXXXXXXX EVCODE #11 is masked for Program Correlation Messages XXX1XXXXXXXXXXXX EVCODE #12 is masked for Program Correlation Messages XX1XXXXXXXXXXXXX EVCODE #13 is masked for Program Correlation Messages X1XXXXXXXXXXXXXX EVCODE #14 is masked for Program Correlation Messages 1XXXXXXXXXXXXXXX EVCODE #15 is masked for Program Correlation Messages

67.4.6 Development Status Register (DS)

The Development Status Register is used to report system debug status. When Debug Mode is entered or exited, or an SoC- or core-defined Low Power Mode is entered (see note below), a Debug Status Message is transmitted with DS[31:24]. The external tool can read this register at any time.



Nexus Reg# - 0x4; DCR - 409; Read-only; Reset - 0x0

Figure 67-8. Development Status Register

Table 67-15. DS field description

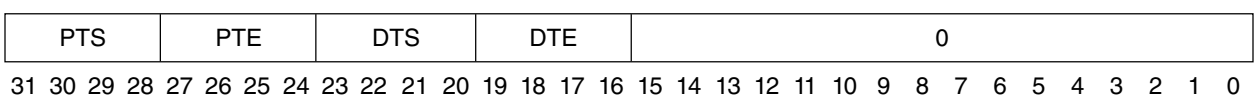
Bits	Name	Description
31	DBG	CPU Debug Mode Status 0 CPU not in Debug mode 1 CPU in Debug mode (**jd_debug_b** signal asserted)
30:28	LPS	System Low Power Mode Status 000 Normal (Run) mode 001 Waiting state (**p_waiting** signal asserted) 010 Halted State (**p_halted** signal asserted) 011 Reserved 1XX Reserved
27:26	LPC	CPU Low Power Mode Status 00 Normal (Run) mode 01 CPU in Halted state (**p_halted** signal asserted) 10 CPU in Stopped state (**p_stopped** signal asserted) 11 CPU in Waiting state (**p_waiting** signal asserted)
25:0	—	Reserved for future functionality (read as 0)

67.4.7 Watchpoint Trigger (WT, PTSTC, PTETC, DTSTC, DTETC) registers

The Watchpoint Trigger Registers allows the watchpoints defined within the Nexus1 logic to trigger actions. These watchpoints can control Program and/or Data Trace enable and disable. The control bits can be used to produce a related "window" for triggering Trace Messages. Watchpoint trigger register WT is used to control triggering by a single selected watchpoint. The Program Trace Start Trigger Control (PTSTC), Program Trace End Trigger Control (PTETC), Data Trace Start Trigger Control (DTSTC), and Data

Trace End Trigger Control (DTETC) are used for extended trigger controls for the respective function. If multiple watchpoints are desired for triggering, or a watchpoint beyond watchpoint #13 is required, then one or more of the extended watchpoint trigger registers may be used. A field encoding of 4'b1111 in one of the WT register fields enables the corresponding extended trigger register. For all other WT field encodings, the corresponding extended trigger register is disabled and the contents are ignored. Note that direct writes to enable program trace and/or data trace in DC1 will override these controls, and trace will remain enabled until another direct write to DC1 to disable program and/or data trace occurs.

When a start trigger is detected, the designated trace features become enabled, and the corresponding enable bits of the DC1 register are set. Whenever a stop trigger is detected, the designated trace features become disabled, and the corresponding enable bits of the DC1 register are cleared. If the same trigger condition is used for both start and stop triggering, then the designated trace features will toggle between being enabled and disabled at each occurrence of the trigger condition. Similarly, if start and stop triggers for a trace feature occur simultaneously, then the designated trace feature will toggle between enabled and disabled depending on the enable state at the time of the trigger events. For example, if tracing is enabled, and a start and stop trigger occur simultaneously, then tracing will be disabled. Direct writes of the DC1 register take precedence over any trace feature enable state that is derived from watchpoint triggering. A table of watchpoints can be found in the Z4 Core Core Debug Support chapter.



Nexus Reg# - 0xB; DCR - 375; Read/Write; Reset - 0x0

Figure 67-9. Watchpoint Trigger (WT) Register

Table 67-16 details the Watchpoint Trigger register fields.

Table 67-16. WT field descriptions

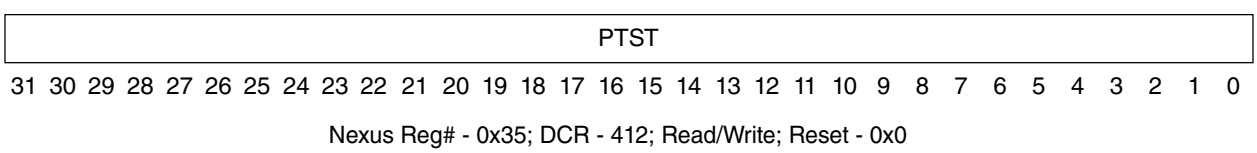
Bits	Name	Description
31:28	PTS	Program Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTSTC register
27:24	PTE	PTE - Program Trace End Control

Table continues on the next page...

Table 67-16. WT field descriptions (continued)

Bits	Name	Description
		0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the PTETC register
23:20	DTS	Data Trace Start Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTSTC register
19:16	DTE	Data Trace End Control 0000 Trigger disabled 0001 Use Watchpoint #0 0010 Use Watchpoint #1 . . 1110 Use Watchpoint #13 1111 Use control settings in the DTETC register
15:0	—	Reserved for future functionality (read as 0)

For extended Program Trace start trigger control, the PTSTC register is used.



Nexus Reg# - 0x35; DCR - 412; Read/Write; Reset - 0x0

Figure 67-10. Program Trace Start Trigger Control (PTSTC) register

Table 67-17 details the PTSTC register fields.

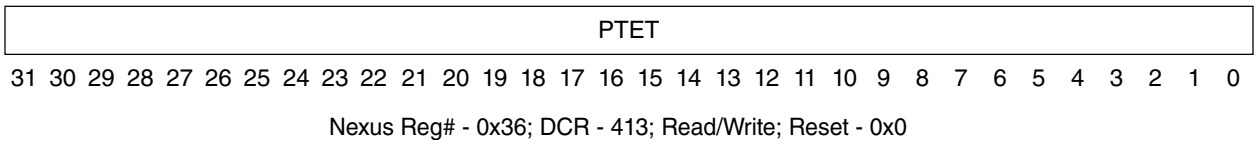
Table 67-17. Program Trace Start Trigger Control Register Fields

Bits	Name	Description
31:0	PTST	Program Trace Start Trigger Control 00000000000000000000000000000000 Trigger disabled

Table 67-17. Program Trace Start Trigger Control Register Fields

Bits	Name	Description
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #30
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Program Trace end trigger control, the PTETC register is used.



Nexus Reg# - 0x36; DCR - 413; Read/Write; Reset - 0x0

Figure 67-11. Program Trace End Trigger Control (PTETC) register

Table 67-18 details the PTETC register fields.

Table 67-18. PTETC field descriptions

Bits	Name	Description
31:0	PTET	PTET - Program Trace End Trigger Control 00000000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19 XXXxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20 XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #21 XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22 XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23 XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24 XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25 XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26 XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27 XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28

Table 67-18. PTETC field descriptions

Bits	Name	Description
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29 X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #30 1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #31

For extended Data Trace start trigger control, the DTSTC register is used.

	DTST

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x37; DCR - 414; Read/Write; Reset - 0x0

Figure 67-12. Data Trace Start Trigger Control (DTSTC) register

The following table details the DTSTC register fields.

Table 67-19. DTSTC field descriptions

Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTST	Data Trace Start Trigger Control 00000000000000000000000000000 Trigger disabled XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0 XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1 XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2 XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3 XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4 XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5 XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6 XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7 XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8 XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9 XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10 XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #11 XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #12 XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #13 XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #14 XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #15 XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #16 XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #17 XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #18

Table 67-19. DTSTC field descriptions

Bits	Name	Description
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XxXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

For extended Data Trace end trigger control, the DTETC register is used.

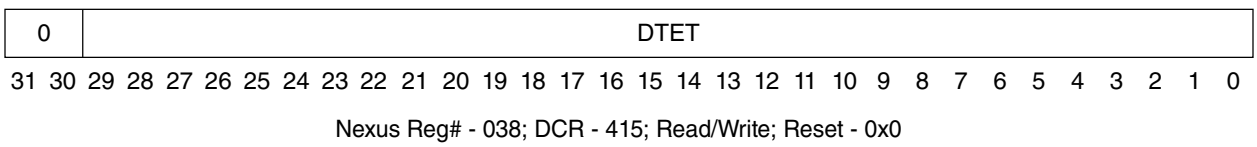


Figure 67-13. Data Trace End Trigger Control (DTETC) register

The following table details the DTETC register fields.

Table 67-20. DTET field descriptions

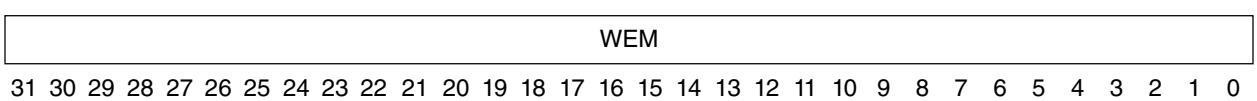
Bits	Name	Description
31:30	—	Reserved for future functionality (read as 0)
29:0	DTET	Data Trace End Trigger Control
		00000000000000000000000000000 Trigger disabled
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Use Watchpoint #0
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Use Watchpoint #1
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Use Watchpoint #2
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Use Watchpoint #3
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Use Watchpoint #4
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Use Watchpoint #5
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Use Watchpoint #6
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Use Watchpoint #7
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Use Watchpoint #8
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Use Watchpoint #9
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #10

Table 67-20. DTET field descriptions

Bits	Name	Description
		XXXXXXXXXXXXXXXXX1XXXXXXXXXX Use Watchpoint #11
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXX Use Watchpoint #12
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXX Use Watchpoint #13
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXX Use Watchpoint #14
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXX Use Watchpoint #15
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXX Use Watchpoint #16
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXX Use Watchpoint #17
		XXXXXXXXXXX1XXXXXXXXXXXXXXXXX Use Watchpoint #18
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXX Use Watchpoint #19
		XxXXXXXXX1XXXXXXXXXXXXXXXXXXX Use Watchpoint #20
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXX Use Watchpoint #21
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXX Use Watchpoint #22
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #23
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #24
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #25
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #26
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #27
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #28
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Use Watchpoint #29

67.4.8 Nexus Watchpoint Mask (WMSK) register

The Nexus Watchpoint Mask register controls which watchpoint events are enabled to produce Watchpoint Trace Messages ($DC1_{TM}$ must also be programmed to generate Watchpoint Trace Messages).



Nexus Reg# - 0x33; DCR - 411; Read/Write; Reset - 0x0

Figure 67-14. Watchpoint Mask Register

The following table details the Watchpoint Trigger register fields.

Table 67-21. WMSK field descriptions

Bits	Name	Description
31:0	WEM	Watchpoint Enable for Messaging

Table continues on the next page...

Table 67-21. WMSK field descriptions (continued)

Bits	Name	Description
		00000000000000000000000000000000 No Watchpoints enabled for Watchpoint Trace Messaging
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1 Watchpoint #0 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX1X Watchpoint #1 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1XX Watchpoint #2 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXXX1XXX Watchpoint #3 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXXX1XXXX Watchpoint #4 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXXX1XXXXX Watchpoint #5 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXXX1XXXXXX Watchpoint #6 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXXX1XXXXXXX Watchpoint #7 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXXX1XXXXXXXX Watchpoint #8 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXXX1XXXXXXXXX Watchpoint #9 enabled for WTM
		XXXXXXXXXXXXXXXXXXXXX1XXXXXXXXXX Watchpoint #10 enabled for WTM
		XXXXXXXXXXXXXXXXXXXX1XXXXXXXXXXX Watchpoint #11 enabled for WTM
		XXXXXXXXXXXXXXXXXXX1XXXXXXXXXXXX Watchpoint #12 enabled for WTM
		XXXXXXXXXXXXXXXXXX1XXXXXXXXXXXXX Watchpoint #13 enabled for WTM
		XXXXXXXXXXXXXXXXX1XXXXXXXXXXXXXX Watchpoint #14 enabled for WTM
		XXXXXXXXXXXXXXXX1XXXXXXXXXXXXXXX Watchpoint #15 enabled for WTM
		XXXXXXXXXXXXXXX1XXXXXXXXXXXXXXXX Watchpoint #16 enabled for WTM
		XXXXXXXXXXXXXX1XXXXXXXXXXXXXXXXX Watchpoint #17 enabled for WTM
		XXXXXXXXXXXXX1XXXXXXXXXXXXXXXXXX Watchpoint #18 enabled for WTM
		XXXXXXXXXXXX1XXXXXXXXXXXXXXXXXXX Watchpoint #19 enabled for WTM
		XXXXXXXXXXX1XXXXXXXXXXXXXXXXXXXX Watchpoint #20 enabled for WTM
		XXXXXXXXXX1XXXXXXXXXXXXXXXXXXXXX Watchpoint #21 enabled for WTM
		XXXXXXXXX1XXXXXXXXXXXXXXXXXXXXXX Watchpoint #22 enabled for WTM
		XXXXXXXX1XXXXXXXXXXXXXXXXXXXXXXX Watchpoint #23 enabled for WTM
		XXXXXXX1XXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #24 enabled for WTM
		XXXXXX1XXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #25 enabled for WTM
		XXXXX1XXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #26 enabled for WTM
		XXXX1XXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #27 enabled for WTM
		XXX1XXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #28 enabled for WTM
		XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #29 enabled for WTM
		X1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #30 enabled for WTM
		1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX Watchpoint #31 enabled for WTM

67.4.9 Nexus Overrun Control Register (OVCR)

The Nexus Overrun Control register controls Nexus behavior as the internal message queues fill up. Response options include suppressing selected message types, or stalling processor instruction execution.

0	SPTHOLD	0	SPEN	0	STTHOLD	0	STEN

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Nexus Reg# - 0x32; DCR - 410; Read/Write; Reset - 0x0

Figure 67-15. Nexus Overrun Control Register

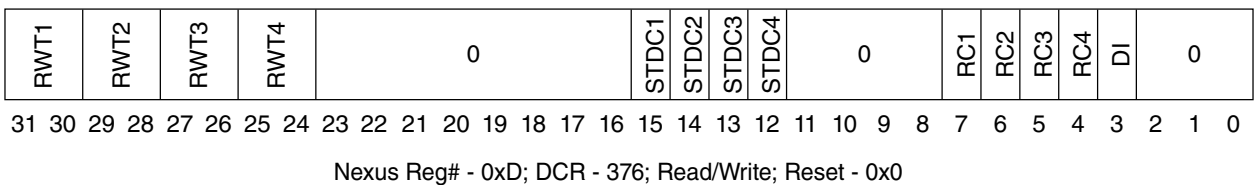
Bits	Name	Description
31:30	—	Reserved, should be cleared
29:28	SPTHOLD	Suppression Threshold 00 - Suppression threshold is when message queues are 1/4 full 01 - Suppression threshold is when message queues are 1/2 full 10 - Suppression threshold is when message queues are 3/4 full 11 - Reserved
27:22	—	Reserved, should be cleared
21:16	SPEN	Suppression Enable 000000 - Suppression is disabled xxxxx1 - Ownership Trace message suppression is enabled xxxx1x - Data Trace message suppression is enabled xxx1xx - Program Trace message suppression is enabled xx1xxx - Watchpoint Trace message suppression is enabled x1xxxx - Reserved 1xxxxx - Data Acquisition message suppression is enabled
15:14	—	Reserved, should be cleared
13:12	STTHOLD	Stall Threshold 00 - Stall threshold is when message queues are 1/4 full 01 - Stall threshold is when message queues are 1/2 full 10 - Stall threshold is when message queues are 3/4 full 11 - Reserved
11:1	—	Reserved, should be cleared
0	STEN	Stall Enable 0 - Stalling is disabled 1 - Stalling is enabled

Figure 67-16. Nexus Overrun Control Register Fields

67.4.10 Data Trace Control Register (DTC) register

The Data Trace Control Register controls whether DTM Messages are restricted to reads, writes, or both for a user programmable address range. In addition, control is provided to restrict data trace message generation to only those load or store accesses that are not stack-related, in order to minimize required data trace message bandwidth.

There are four Data Trace channels controlled by the DTC for the Nexus 3 module. Channels can be programmed to trace data accesses or instruction accesses, but not independently.



Nexus Reg# - 0xD; DCR - 376; Read/Write; Reset - 0x0

Figure 67-17. Data Trace Control Register

The following table details the Data Trace Control register fields.

Table 67-22. DTC field descriptions

Bits	Name	Description
31:30	RWT1	Read/Write Trace 1 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
29:28	RWT2	Read/Write Trace 2 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
27:26	RWT3	Read/Write Trace 3 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
25:24	RWT4	Read/Write Trace 4 00 No trace enabled X1 Enable Data Read Trace 1X Enable Data Write Trace
23:16	—	Reserved for future functionality (read as 0)
15	STDC1	Stack Trace Disable Control 1 0 Tracing of stack accesses are not disabled for range 1

Table continues on the next page...

Table 67-22. DTC field descriptions (continued)

Bits	Name	Description
		1 Tracing of stack accesses are disabled for range 1; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
14	STDC2	Stack Trace Disable Control 2 0 Tracing of stack accesses are not disabled for range 2 1 Tracing of stack accesses are disabled for range 2; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
13	STDC3	Stack Trace Disable Control 3 0 Tracing of stack accesses are not disabled for range 3 1 Tracing of stack accesses are disabled for range 3; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
12	STDC4	Stack Trace Disable Control 4 0 Tracing of stack accesses are not disabled for range 4 1 Tracing of stack accesses are disabled for range 4; data accesses using GPR R1 in effective address computations are not traced and do not generate data range watchpoint outputs
11:8	—	Reserved for future functionality (read as 0)
7	RC1	Range Control 1 0 Condition trace on address within range 1 Condition trace on address outside of range
6	RC2	Range Control 2 0 Condition trace on address within range 1 Condition trace on address outside of range
5	RC3	Range Control 3 0 Condition trace on address within range 1 Condition trace on address outside of range
4	RC4	Range Control 4 0 Condition trace on address within range 1 Condition trace on address outside of range
3	DI	Data Access / Instruction Access Trace 0 Condition trace on data accesses 1 Condition trace on instruction accesses The support for monitoring instruction accesses is not guaranteed to be present in all versions of the Nexus 3 module. The setting of the DI bit also affects the information provided on the data trace port outputs (See the "Data Trace Port Signals" section in the Z4 Core Core Debug Support chapter.)
2:0	—	Reserved for future functionality (read as 0)

67.4.11 Data Trace Start Address Registers (DTSA1-4)

The Data Trace Start Address Registers define the start addresses for each trace channel.

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0xE; DCR - 377; Read/Write; Reset - 0x0

Figure 67-18. Data Trace Start Address 1 (DTSA1) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0xF; DCR - 378; Read/Write; Reset - 0x0

Figure 67-19. Data Trace Start Address 2 (DTSA2) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x10; DCR - 379; Read/Write; Reset - 0x0

Figure 67-20. Data Trace Start Address 3 (DTSA3) register

Data Trace Start Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x11; DCR - 380; Read/Write; Reset - 0x0

Figure 67-21. Data Trace Start Address 4 (DTSA4) register

67.4.12 Data Trace End Address (DTEA1-4) registers

The Data Trace End Address Registers define the end addresses for each trace channel.

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x12; DCR - 381; Read/Write; Reset - 0x0

Figure 67-22. Data Trace End Address 1 (DTEA1) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x13; DCR - 382; Read/Write; Reset - 0x0

Figure 67-23. Data Trace End Address 2 (DTEA2) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x14; DCR - 383; Read/Write; Reset - 0x0

Figure 67-24. Data Trace End Address 3 (DTEA3) register

Data Trace End Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Nexus Reg# - 0x15; DCR - 408; Read/Write; Reset - 0x0

Figure 67-25. Data Trace End Address 4 (DTEA4) register

Table 67-23 illustrates the range that will be selected for Data Trace for various cases of DTSA being less than, greater than, or equal to DTEA.

Table 67-23. Data trace - address range options

Programmed values	Range Control Bit Value	Range Selected	
DTSA < DTEA	0	DTSA ->	<- DTEA
DTSA < DTEA	1	<- DTSA	DTEA ->
DTSA > DTEA	N/A	Invalid Range - no trace	
DTSA = DTEA	N/A	Invalid Range - no trace	

Note

DTSA must be less than DTEA in order to guarantee correct Data Write/Read Traces. Data Trace ranges are *inclusive* of the DTSA and DTEA addresses for Range Control settings indicating "within range," and are *exclusive* of the DTSA and DTEA addresses for Range Control settings indicating "outside of range."

Accesses that meet the range and access type qualifiers will cause assertion of a watchpoint output for Ranges 1, 2, and 3. There are three dedicated watchpoint outputs, one for each DTSA/DTEA sets 1, 2, and 3. Range 4 does not provide a watchpoint output. Note that when DTC_{DI}=1, all instruction fetches and prefetches (including discarded prefetches) are monitored, and thus these range watchpoints differ from the IACx watchpoint outputs, which are not asserted for instructions that are not executed (i.e. when the instruction prefetch is discarded).

67.4.13 Read/Write Access Control/Status (RWCS) register

The Read Write Access Control/Status Register provides control for Read/Write Access. Read/Write access provides DMA-like access to memory-mapped resources on the AHB System bus either while the processor is halted, or during runtime. Control is provided over access type, size, count, and certain bus attributes.

Note that the internal CPU interfaces to the caches and local memory are not accessible or utilized by this mechanism. Since the external interface is used, base address port settings are used to access local memory, and not the settings of local memory control register base address values.

The RWCS Register also provides Read/Write Access Status information per Table 67-25.

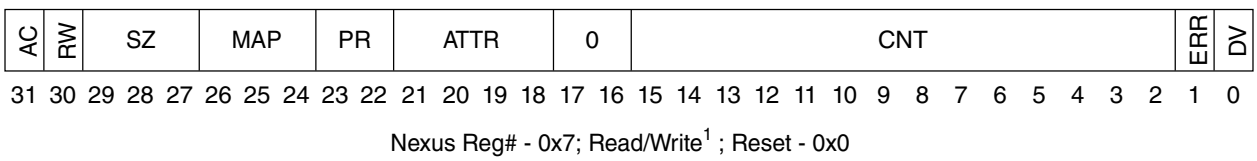


Figure 67-26. Read/Write Access Control/Status (RWCS) register

NOTES:

ERR and DV are read-only.

Table 67-24. RWCS field description

Bits	Name	Description
RWCS[31]	AC	Access Control 0 End access/ Access has completed 1 Start access
RWCS[30]	RW	Read/Write Select 0 Read access 1 Write access
RWCS[29:27]	SZ	Word Size 000 8-bit (byte) 001 16-bit (half-word) 010 32-bit (word) 011 64-bit (doubleword, requires two passes through RWD) 100 - 111 Reserved (default to word)
RWCS[26:24]	MAP	MAP Select 000 Primary memory map 001-111 Reserved
RWCS[23:22]	PR[1]	Read/Write Access Priority

Table continues on the next page...

Table 67-24. RWCS field description (continued)

Bits	Name	Description
		00 Reserved (default to highest priority) 01 Reserved (default to highest priority) 10 Reserved (default to highest priority) 11 Highest access priority
RWCS[21:18]	ATTR	Access Attributes 0xxx Reserved 1xxx Reserved x0xx **p_d_hprot[4]** driven to 0 for accesses x1xx **p_d_hprot[4]** driven to 1 for accesses xx0x **p_d_hprot[3]** driven to 0 for accesses xx1x **p_d_hprot[3]** driven to 1 for accesses xxx0 **p_d_hprot[2]** driven to 0 for accesses xxx1 **p_d_hprot[2]** driven to 1 for accesses
RWCS[17:16]	—	RES - Reserved for future functionality
RWCS[15:2]	CNT	Access Control Count hhhh Number of accesses of word size SZ
RWCS[1]	ERR	Read/Write Access Error (see Table 67-25)
RWCS[0]	DV[2]	Read/Write Access Data Valid (see Table 67-25)

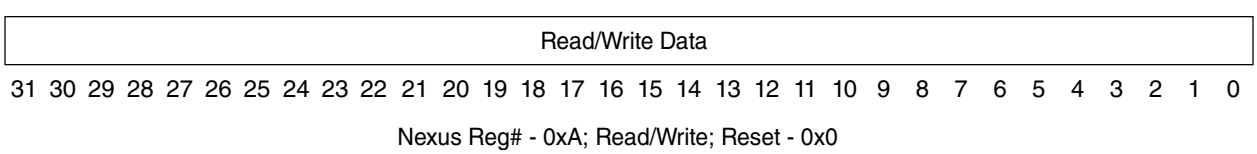
1. The priority functionality is not currently implemented.
2. ERR and DV are read-only.

Table 67-25. Read/Write Access Status Bit Encoding

Read Action	Write Action	ERR	DV
Read Access has not completed	Write Access completed without error	0	0
Read Access error has occurred	Write Access error has occurred	1	0
Read Access completed without error	Write Access has not completed	0	1
Not Allowed	Not allowed	1	1

67.4.14 Read/Write Access Data (RWD) register

The Read/Write Access Data (RWD) register provides the data to/from system bus memory-mapped locations when initiating a read or a write access.



Nexus Reg# - 0xA; Read/Write; Reset - 0x0

Figure 67-27. Read/Write Access Data (RWD) register

Read/Write accesses to the AHB require that the debug firmware properly retrieve/place the data in the RWD. Table 67-26 shows the proper placement of data into the RWD. Note that doubleword transfers require two passes through RWD.

Table 67-26. RWD data placement for ransfers

Transfer Size and byte offset	RWA(2:0)	RWCS[SZ]	RWD			
			31:24	23:16	15:8	7:0
Byte	x x x	0 0 0	-	-	-	X
Half	x x 0	0 0 1	-	-	X	X
Word	x 0 0	0 1 0	X	X	X	X
Doubleword	0 0 0	0 1 1				
first RWD pass (low order data)			X	X	X	X
second RWD pass (high order data)			X	X	X	X

Table Notes:

"X" indicates byte lanes with valid data

"-" indicates byte lanes that will contain unused data.

Table 67-27 shows the mapping of RWD bytes to byte lanes of the AHB read and write data buses.

Table 67-27. RWD byte lane mapping

Transfer Size and byte offset	RWA(2:0)	RWD			
		31:24	23:16	15:8	7:0
Byte @000	0 0 0	-	-	-	AHB[7:0]
Byte @001	0 0 1	-	-	-	AHB[15:8]
Byte @010	0 1 0	-	-	-	AHB[23:16]
Byte @011	0 1 1	-	-	-	AHB[31:24]
Byte @100	1 0 0	-	-	-	AHB[39:32]
Byte @101	1 0 1	-	-	-	AHB[47:40]
Byte @110	1 1 0	-	-	-	AHB[55:48]
Byte @111	1 1 1	-	-	-	AHB[63:56]
Half @000	0 0 0	-	-	AHB[15:8]	AHB[7:0]
Half @010	0 1 0	-	-	AHB[31:24]	AHB[23:16]
Half @100	1 0 0	-	-	AHB[47:40]	AHB[39:32]
Half @110	1 1 0	-	-	AHB[63:56]	AHB[55:48]
Word @000	0 0 0	AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
Word @100	1 0 0	AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]
Doubleword @000	0 0 0				
first RWD pass		AHB[31:24]	AHB[23:16]	AHB[15:8]	AHB[7:0]
second RWD pass		AHB[63:56]	AHB[55:48]	AHB[47:40]	AHB[39:32]

Table Notes:

"-" indicates byte lanes that will contain unused data.

67.4.15 Read/Write Access Address (RWA)

The Read/Write Access Address Register provides the system bus address to be accessed when initiating a read or a write access.

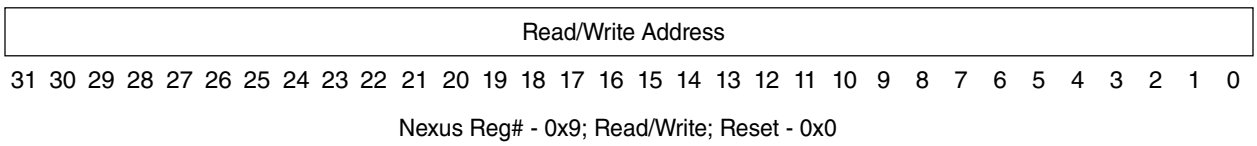


Figure 67-28. Read/Write Access Address (RWA) register

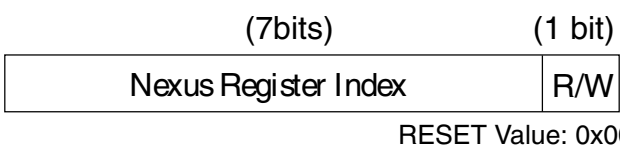
67.5 Nexus 3 Register Access via JTAG/OnCE

Access to Nexus 3 register resources is enabled by loading a single instruction ("*NEXUS3-ACCESS*") into the JTAG Instruction Register (IR) (OnCE OCMD register). For the Nexus 3 block, the OCMD value is 0b0001111100.

Once the NEXUS3-ACCESS instruction has been loaded, the JTAG/OnCE port allows tool/target communications with all Nexus 3 registers according to the register map in Table 67-8.

Reading/writing of a Nexus 3 register then requires two (2) passes through the Data-Scan (DR) path of the JTAG state machine. (See IEEE 1149.1 (JTAG) RD/WR sequences.)

1. The first pass through the DR selects the Nexus 3 register to be accessed by providing an index (see Table 67-8), and the direction (read/write). This is achieved by loading an 8-bit value into the JTAG Data Register (DR). This register has the following format:



Nexus Register Index	Selected from values in Table 67-8
Read/Write (R/W):	0 - Read 1 - Write

2. The second pass through the DR then shifts the data in or out of the JTAG port, LSB first.
    a. During a read access, data is latched from the selected Nexus register when the JTAG state machine passes through the "Capture-DR" state.
    b. During a write access, data is latched into the selected Nexus register when the JTAG state machine passes through the "Update-DR" state.

67.6 Nexus 3 Register Access via Software

Access to most Nexus 3 register resources by software is supported by mapping the Nexus 3 registers to privileged DCRs (device control registers) that are accessible via the **mfdcr** and **mtdcr** instructions. It is intended that these access only occur when no possible conflicts with hardware-based accesses are possible. A conflict between hardware and software accesses will produce undefined results. Table 67-8 shows the mappings of Nexus 3 registers to DCR numbers. Software writes to Nexus 3 register resources are blocked when the **nex_sfwcntl_en** input signal is negated, without signaling an exception. Note that the Nexus 3 Read/Write Access functionality is not available to software, since software can use load and store instructions to access memory.

67.7 Nexus Message Fields

Nexus messages are comprised of fields. Each field contains a distinct piece of information within a message, and each message contains multiple fields. Messages are transferred in packets over the Auxiliary Output protocol. A packet is a collection of fields. A packet may contain any number of fixed length fields, but may contain at most one variable length field. The variable length field must be the last field in a packet. The following sub-sections describe a subset of the message field types.

67.7.1 TCODE Field

The TCODE field is a 6-bit fixed length field that identifies the type of message and its format. The field encodings are assigned by IEEE-ISTO 5001.

67.7.2 Source ID Field (SRC)

Each Nexus module in a device is identified by a unique Client Source Identification Number. The number assigned to each Nexus module is determined by the SoC integrator, and is provided on the **nex3_ext_src_id[0:3]** input signals. Multi-threaded processors may assign additional source ID information to indicate which thread a message is associated with. The Nexus 3 module implements a 4-bit fixed length Source ID field consisting of a Client Source ID.

67.7.3 Relative Address Field (U-ADDR)

The non-sync forms of the Program and Data Trace messages include addresses that are relative to the address that was transmitted in the previous Program or Data Trace message, respectively. The relative address format is compliant with IEEE-ISTO 5001 and is designed to reduce the number of bits transmitted for address fields.

The relative address is generated by XORing the new address with the previous, and then using only the results up to the most significant '1'. To recreate the original address, the relative address is XORed with the previously decoded address.

The relative address of a Program Trace message is calculated with respect to the previous Program Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Program Trace messages.

The relative address of a Data Trace message is calculated with respect to the previous Data Trace message, regardless of any address information that may have been sent in any other trace messages in the interim between the two Data Trace messages.

Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

Previous Address (A1) =0x0003FC01, New Address (A2) = 0x0003F365

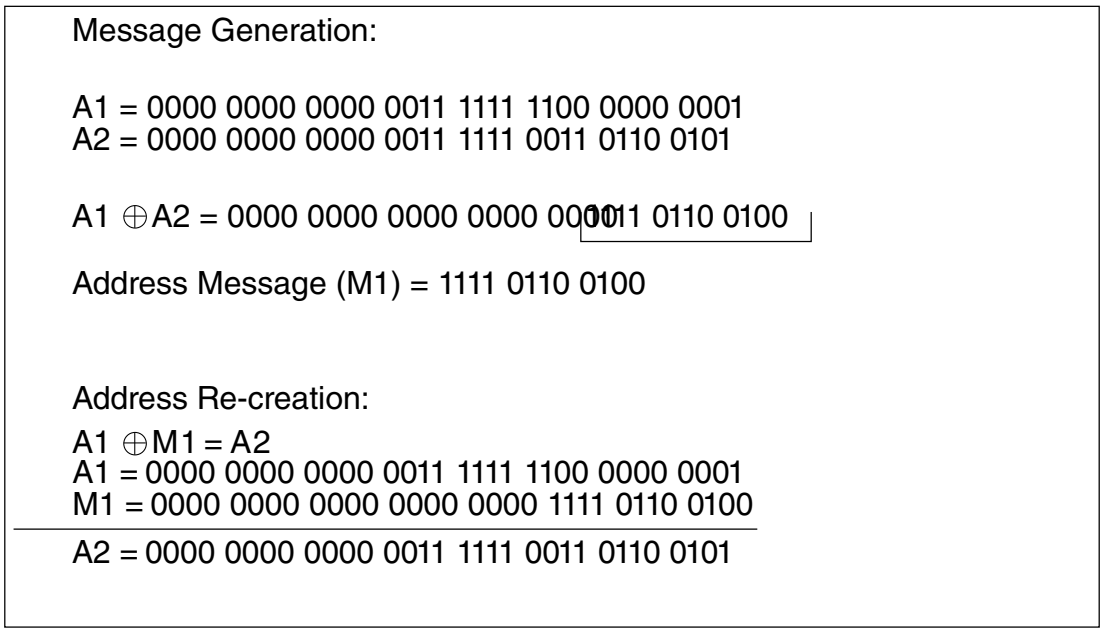


Figure 67-29. Relative address generation and recreation

67.7.4 Full Address Field (F-ADDR)

Program Trace synchronization messages provide the full address associated with the trace event (leading zeroes may be truncated) with the intent of providing a reference point for development tools to operate from when reconstructing relative addresses. Synchronization messages are generated at significant mode switches and are also generated periodically to ensure that development tools are guaranteed to have a reference address given a sufficiently large sample of trace messages. Program trace messages also provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

67.8 Nexus Message Queues

The Nexus 3 module implements internal message queues capable of storing up to three messages per cycle into a small initial queue, which then fills a larger queue at up to two messages per cycle. Messages that enter the queues are transmitted in the order in which they are received.

If more than three messages attempt to enter the queue in the same cycle, the highest priority messages are stored and the remaining message(s) will be dropped due to a collision. Collision events are expected to be rare.

The Overrun Control register (OVCR) controls the Nexus behavior as the message queue fills. The Nexus block may be programmed to:

- Allow the queue to overflow, drain the contents, queue an overrun error message and resume tracing.
- Stall the processor when the queue utilization reaches the selected threshold.
- Suppress selected message types when the queue utilization reaches the selected threshold.

67.8.1 Message Queue Overrun

In this mode, the message queue will stop accepting messages when an overrun condition is detected. The contents of the queues will be allowed to drain until empty. Incoming messages are discarded until the queue is emptied. Once empty, an overrun error message is enqueued that contains information about the types of messages that were discarded due to the overrun condition.

67.8.2 CPU Stall

In this mode, processor instruction issue is stalled when the queue utilization reaches the selected threshold. The processor is stalled long enough to drop one threshold level below the level that triggered the stall. For example, if stalling the processor is triggered at 1/4 full, the stall will stay in effect until the queue utilization drops to empty. There may be significant skid from the time that the stall request is made until the processor is able to stop completing instructions. This skid should be taken into consideration when programming the threshold. Refer to Nexus Overrun Control Register (OVCR) for complete programming options.

67.8.3 Message Suppression

In this mode, the message queue will disable selected message types when the queue utilization reaches the selected threshold. This allows lower bandwidth tracing to continue and possibly avoid an overrun condition. If an overrun condition occurs despite this message suppression, the queue will respond according to the behavior described in

Message Queue Overrun. Suppressed message types are reported in the error message if they occur after overrun in addition to other discarded message types. Once triggered, message suppression will remain in effect until queue utilization drops to the threshold below the level selected to trigger suppression.

67.8.4 Nexus Message Priority

Nexus messages may be lost due to contention with other message types under the following circumstances:

- More than three messages are generated in the same cycle

Table 67-28 lists the various message types and their relative priority from highest to lowest.

Up to three message requests can be queued into the message buffer in a given cycle. If more than three message requests exist in a given cycle, the three highest priority message classes are queued into the message buffer. The remaining messages that did not successfully queue into the message buffer in that cycle will generate subsequent responses, as detailed in Table 67-28.

The CPU is capable of completing two instructions per cycle. If multiple trace messages need to be queued at the same time, they will be queued with the following priority: Instruction 0 (oldest instruction) (WPM ->DQM -> PCM_{PIDMSG} -> OTM -> BTM -> DTM)-> Instruction1 (newer instruction) (WPM -> DQM -> OTM -> BTM -> DTM). As many as three messages may be simultaneously queued. Note that for the cycle following a dropped PTM, non-periodic OTM, or DQM message, only two other messages may be queued in addition to the dropped Error Message.

Watchpoint Messages from instructions that complete at the same time or events that occur during the same cycle will be combined.

Table 67-28. Message Type Priority and Message Dropped Responses

Message Type	Message	Priority	Message Dropped Response
Error	Error	0 (highest)	N/A[1]
WP (Watchpoint Trace)	WPM (Watchpoint Message)	1	N/A[1]
DQ (Data Acquisition)	DQM (Data Acquisition Message)	2	DQM Error Message
Program Trace (PID MSG)	PCM - PID/NPIDR update (Program Correlation Message)	2	OTM Error Message

Table continues on the next page...

Table 67-28. Message Type Priority and Message Dropped Responses (continued)

Message Type	Message	Priority	Message Dropped Response
OT (Ownership)	OTM - PID/NPIDR update (Ownership Trace Message)	2	OTM Error Message[2]
Program Trace	BTM (Branch Trace Message)	2	BTM Error Message, Sync upgrade next BTM
	Sync (Program Sync Message)	3	Sync upgrade next BTM
	RFM (Resource Full for Instruction counter or history buffer)	4	BTM Error Message Sync upgrade next BTM
	DS (Debug Status Message)	5	Sync upgrade next BTM
	PCM (Program Correlation Message)	6	BTM Error Message Sync upgrade next BTM
DT (Data Trace)	DTM (Data Trace Message)	7	Sync upgrade next DTM
OT (Ownership)	OTM - Periodic update (Ownership Trace Message)	8 (lowest)	none

1. Error and Watchpoint messages are not dropped due to collisions, due to their priority.
2. Message will always be dropped if program trace is enabled, and program correlation messages for PID messages are not masked (Event Code = 0101). No error message is sent for this case since the PID value is contained in the higher priority message.

67.8.5 Data Acquisition Message Priority Loss Response

If a Data Acquisition Message (DQM) loses arbitration due to contention with higher priority messages, an error message will be generated to indicate that a DQM has been lost due to contention.

67.8.6 Ownership Trace Message Priority Loss Response

If an Ownership Trace message (OTM) due to software updates to the Process ID state loses arbitration due to contention with higher priority messages other than a program correlation message with EVCODE = 0101 (PID update), an error message will be generated to indicate that a OTM has been lost due to contention. If the pending OTM is a periodic update, the event is dropped without generating an error message.

67.8.7 Program Trace Message Priority Loss Response

If a Program Trace message (PTM) loses arbitration due to contention with higher priority messages, and the discarded PTM is a Program Correlation message, a Resource Full message for instruction count or history buffer, or a Branch Trace message, then an Error message is generated to indicate that branch trace information has been lost, and the next Branch Trace message will be upgraded to a sync-type message.

If the discarded PTM is a Program Correlation message with PID information (EVCODE=0101), the Error message will indicate a dropped OTM and a dropped Program Trace (Error code = xxxx11xx).

67.8.8 Program Trace Sync Message Priority Loss Response

If a Program Trace Sync message (SYNC) loses arbitration due to contention with higher priority messages, the Sync event is discarded, and the next Branch Trace message will be upgraded to a sync-type message.

67.8.9 Data Trace Message Priority Loss Response

If a Data Trace message (DTM) loses arbitration due to contention with higher priority messages, the DTM event is discarded, and the next DTM will be upgraded to a sync-type message.

67.9 Debug Status messages

Debug Status Messages report low power mode and debug status. Debug Status messages are enabled when Nexus 3 is enabled. Entering/exiting Debug Mode as well as entering, exiting, or changing Low Power Mode(s) will trigger a Debug Status message, indicating the value of the most significant byte in the Development Status register. Debug status information is sent out in the following format:

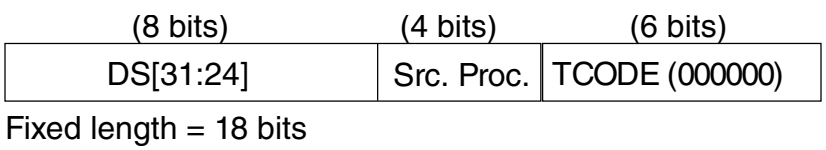


Figure 67-30. Debug Status message format

67.10 Error messages

Error messages are enabled whenever the debug logic is enabled. There are two conditions that will produce an error message, each receiving a separate error type designation:

- A message is discarded due to contention with other (higher priority) message types. These errors will have an Error Type value of 1.
- The message queue overruns. After the queue is drained, an error message is enqueued with an error code that indicates what types of messages were discarded during the interim. These errors will have an Error Type value of 0.

Note

The OVCR Register can be used in order to alleviate potential overrun situations.

Error information is messaged out in the following format (also see Table 67-3 and Table 67-4):

(6 bits)	(4 bits)	(4 bits)	(6 bits)
Error Code	Error Type	Src. Proc.	TCODE (001000)

Fixed length = 20 bits

Figure 67-31. Error message format

67.11 Ownership trace

This section details the ownership trace features of the Nexus 3 module.

67.11.1 Overview

Ownership trace provides a macroscopic view, such as task flow reconstruction, when debugging software written in a high level (or object-oriented) language. It offers the highest level of abstraction for tracking operating system software execution. This is especially useful when the developer is not interested in debugging at lower levels.

67.11.2 Ownership Trace Messaging (OTM)

Ownership trace information is messaged via the auxiliary port using an Ownership Trace Message (OTM). Z4 Core processors contain a *PowerISA 2.06* defined "Process ID" register within the CPU. It is updated by the operating system software to provide task/process ID information. The contents of this register are replicated on the pins of the processor and connected to Nexus. The Process ID (PID) register value can be accessed using the **mfspr/mtspr** instructions.

Note

The CPU includes a Process ID register (PID0), thus the Nexus UBA functionality is not implemented.

In addition, to decouple trace information from the PID0 register and to provide a full independent 32-bit process ID for debug use, the Nexus PID Register (NPIDR) may be used instead of PID0 to provide OTM process ID information. It is updated by the operating system software to provide task/process ID information. The Nexus Process ID (NPIDR) register value can be accessed using the **mfspr/mtspr** instructions. This register is accessible in user or supervisor mode.

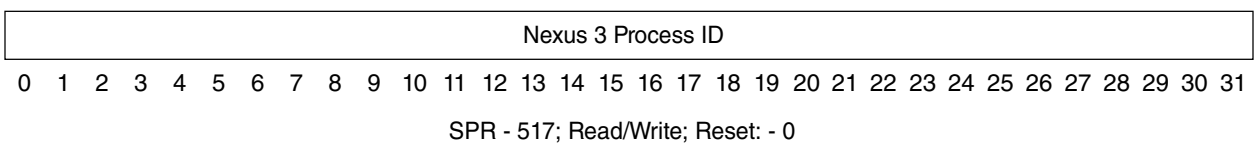


Figure 67-32. Nexus 3 Process ID Register (NPIDR)

Note

OS updates to NPIDR should be performed in addition to normal PID0 updates when a process switch occurs, in order to properly generate OTM messages with new process ID information when NPIDR is selected for OTM use.

The process ID source (PID0 or NPIDR) is selected by the setting of the $DC1_{OTS}$ control bit.

There are two conditions that will cause an Ownership Trace Message when Ownership Trace is enabled:

- When the PID0 register is written to by the processor and $DC1_{OTS}$ indicates PID0 should be used, or when the NPIDR register is written to by the processor and $DC1_{OTS}$ indicates NPIDR should be used, the data is latched within Nexus, and is messaged out via the auxiliary port, allowing development tools to trace ownership flow. However, if Program Trace is enabled, and program correlation messages for

PID updates are not masked (Event Code = 0101), then an OTM will not be generated for an update to the selected process ID register, since the program correlation message will provide this process ID update information.

- Periodically, at least once every 256 messages, the most recent state of the selected process ID register is messaged out. The resulting Ownership Trace message will indicate in the PID Index sub-field that PID0/NPIDR status is being reported and the most recent value of the PID0/NPIDR register will be conveyed in the Process ID value sub-field. These periodic Ownership Trace message events can be disabled by setting $DC1_{POTD}$.

Ownership trace information is messaged out in the following format:

(1-32 bits)	(4 bits)	(4 bits)	(6 bits)
Process ID	PID Index (0000)	Src. Proc.	TCODE (000010)

Variable length = 15-46 bits

Figure 67-33. Ownership Trace Message format

67.12 Program trace

This section details the program trace mechanism supported by Nexus3 for the z4 processor. Program trace is implemented via Branch Trace Messaging (BTM) as per the **IEEE-ISTO 5001** standard definition. Branch Trace Messaging for z4 processors is accomplished by snooping the internal address bus (between the CPU and Cache), attribute signals, and CPU Status (**p_mode[0:3]**, **p_pstat_pipe{0,1}[0:5]**).

67.12.1 Branch Trace Messaging types

Traditional Branch Trace Messaging facilitates program trace by providing the following types of information:

- Messaging for taken direct branches includes how many sequential instructions were executed since the last taken branch or exception, including the taken direct branch. Branch instructions are included in the count of sequential instructions.
- Messaging for taken indirect branches and exceptions includes how many sequential instructions were executed since the last taken branch or exception and the unique portion of the branch target address or exception vector address. Branch instructions are included in the count of sequential instructions. For taken indirect branches that trigger generation of a message, the branch is also included in the count.

Branch History Messaging facilitates program trace by providing the following information.

- Messaging for taken indirect branches and exceptions includes a) how many sequential instructions (I-CNT) were executed since the last predicate instruction, taken/not taken direct branch, taken/not-taken indirect branch, or exception, b) the unique portion of the branch target address or exception vector address, and c) a branch/predicate instruction history field. Each bit in the history field represents a direct branch or predicated instruction where a value of one (1) indicates taken, and a value of zero (0) indicates not taken. Not-taken indirect branches will generate a history bit with a value of zero (0). Instructions that generate history bits are not included in instruction counts. For taken indirect branches that trigger generation of this message type, the branch is included in the count, but not in the history field.

67.12.1.1 Indirect Branch Message instructions

Table 67-29 shows the types of instructions and events that cause Indirect Branch Messages or Branch History Messages to be encoded.

Table 67-29. Indirect Branch Message sources

Source of Indirect Branch Message	Instructions / Detail
Taken branch relative to a register value	**bcctr, bcctrl, bclr, bclrl, se_bctr, se_bctrl, se_blr, se_blrl**
System Call / Trap exceptions taken	**se_sc, tw**
Return from interrupts / exceptions	**se_rfi, se_rfci, se_rfdi**
Exit from reset with Program Trace Enabled	**Indirect branch with Sync, target address is initial instruction, count = 1**

67.12.1.2 Direct Branch Message Instructions

Table 67-30 shows the types of instructions that will cause Direct Branch Messages or will toggle a bit in the instruction history buffer to be messaged out in a Resource Full Message or Branch History Message.

Table 67-30. Direct Branch Message sources

Source of Direct Branch Message	Instructions
Taken direct branch instructions Instruction Synchronize	**b, ba, bl, bla, bc, bca, bcl, bcla, se_b. se_bc, se_bl, e_b, e_bc, e_bl, e_bcl, se_isync**

67.12.1.3 BTM using Branch History Messages

Traditional BTM Messaging can accurately track the number of sequential instructions between branches, but cannot accurately indicate which instructions were conditionally executed, and which were not.

Branch History Messaging solves this problem by providing a predicated instruction history field in each Indirect Branch Message. Each bit in the history represents a predicated instruction or direct branch, or a not-taken indirect branch. A value of one (1) indicates the conditional instruction was executed or the direct branch was taken. A value of zero (0) indicates the conditional instruction was not executed or the branch was not taken.

Branch History Messages solve predicated instruction tracking and save bandwidth since only indirect branches cause messages to be queued.

67.12.1.4 BTM using Traditional Program Trace Messages

Based on the PTM bit in the DC1 Register, Program Tracing can utilize either Branch History Messages (PTM = 1) or traditional Direct/Indirect Branch Messages (PTM = 0).

Branch History will save bandwidth and keep consistency between methods of Program Trace, yet may lose temporal order between BTM messages and other types of messages. Since direct branches are not messaged, but are instead included in the history field of the Indirect Branch History Message, other types of messages may enter the FIFO between Branch History Messages. The development tool cannot determine the ordering of "events" that occurred with respect to direct branches simply by the order in which messages are sent out.

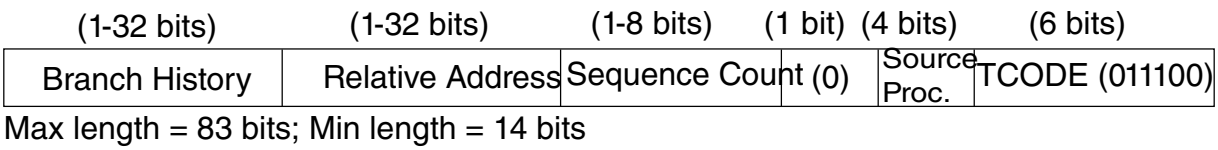
Traditional BTM messages maintain their temporal ordering because each event that can cause a message to be queued will enter the FIFO in the order it occurred and will be messaged out maintaining that order.

67.12.2 BTM Message For mats

The Nexus 3 block supports three types of traditional BTM Messages - Direct, Indirect, and Synchronization Messages. It supports two types of branch history BTM Messages - Indirect Branch History, and Indirect Branch History with Synchronization Messages.

67.12.2.1 Indirect branch messages (history)

Indirect branches include all taken branches whose destination is determined at run time, interrupts, and exceptions. If $DC1_{PTM}$ is set to '1', indirect branch information is messaged out in the following format:

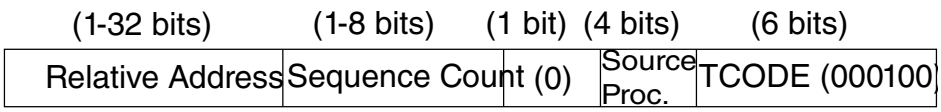


Max length = 83 bits; Min length = 14 bits

Figure 67-34. Indirect branch message (history) format

67.12.2.2 Indirect branch messages (traditional)

If $DC1_{PTM}$ is cleared to '0', indirect branch information is messaged out in the following format:



Max length = 51 bits; Min length = 13 bits

Figure 67-35. Indirect branch message format

67.12.2.3 Direct branch messages (traditional)

Direct branches (conditional or unconditional) are all taken branches whose destination is fixed in the instruction opcode. Direct branch information is messaged out in the following format:

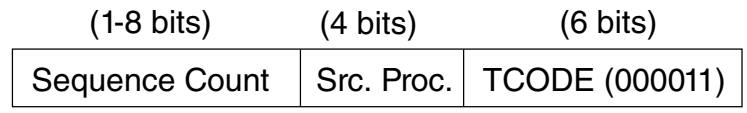


Figure 67-36. Direct Branch message format

Note

When $DC1_{PTM}$ is set, Direct Branch Messages will not be transmitted. Instead, each direct branch, not-taken indirect branch, or predicated instruction will be recorded in the history buffer.

67.12.3 Program Trace Message Fields

The following subsections describe specific fields used for Program Trace messages.

67.12.3.1 Sequential Instruction Count Field (ICNT)

Most of the program trace messages include an instruction count field. For traditional Branch messages, ICNT represents the number of sequential instructions including non-taken branches since the last Direct/Indirect Branch messages. Branch instructions that trigger message generation are included in the ICNT.

For Branch History messages, ICNT represents the number of instructions executed since the last taken/non-taken direct branch, predicate instruction, last taken/not-taken indirect branch, or exception. Branch instructions that trigger message generation are included in the ICNT. Instructions that generate history bits are not included in the ICNT.

The sequential instruction counter overflows after its value reaches 255 and is reset to 0. In addition, the next BTM Message (corresponding to the 256th or later instruction) will be converted to a w/sync type message.

The instruction counter is reset every time the instruction count is transmitted in a message or whenever there is a branch/predicate history event, as well as on exiting from debug mode.

67.12.3.2 Branch/Predicate Instruction History (HIST)

If $DC1_{PTM}$ is set, BTM messaging will use the Branch History format. The branch history (HIST) field in these messages provides a history of branch execution used for reconstructing the program flow. The branch/predicate history buffer stores information about branch and predicate instruction execution. The buffer is implemented as a left-shifting register. The buffer is preloaded with a one (1), which acts as a stop bit (the most significant 1 in the history field is a termination bit for the field). The pre-loaded bit itself is not part of the history, but is transmitted with the packet.

A value of one (1) is shifted into the history buffer for each taken direct branch (program counter relative branch) or predicate instruction whose condition evaluates to true. A value of zero (0) is shifted into the history buffer for each not-taken branch (including indirect branch instructions) or predicate instruction whose condition evaluates to false.

This history buffer information is transmitted as part of an Indirect Branch with History message, as part of a Program Correlation message, or as part of a Resource Full message if the history buffer becomes full. The history buffer is reset every time the history information is transmitted in a message, as well as on exiting from debug mode.

Table 67-31. Branch/Predicate history events

Branch/Predicate History Event	History Bit(s)	Relevant Instructions
Not taken register indirect branches	0	**bcctr, bcctrl, bclr, bclrl**
Not taken direct branches	0	**b, ba, bc, bca, bla, bcla, bl, bcl**
Taken direct branches	1	**b, ba, bc, bca, bla, bcla, bl, bcl**[1]

1. If the EVCODE for direct branch function calls is not masked in DC4, taken **bl** and **bcl** instructions will generate Program Correlation Messages and will not be logged in the history buffer.1

67.12.3.3 Execution Mode Indication

In order for a development tool to properly interpret instruction count and history information, it must be aware of the execution mode context of that information. VLE instructions will be interpreted differently from non-VLE instructions.

Program trace messages provide the execution mode status in the least significant bit of the **reconstructed** address field. A value of '0' indicates that preceding instruction count and history information should be interpreted in a non-VLE context. A value of '1' indicates that the preceding instruction count and history information should be interpreted in a VLE context.

67.12.4 Resource Full Messages

The Resource Full Message is used in conjunction with Branch Trace and Branch History Messages. The Resource Full Message is generated when either the internal branch/ predicate history buffer is full, or if the BTM Instruction sequence counter (I-CNT) overflows. If synchronization is needed at the time this message is generated, the synchronization is delayed until the next Branch Trace Message that is not a Resource Full Message.

For history buffer overflow, the Resource Full Message transmits a Resource Code (RCODE) of 0b0001 and the current contents of the history buffer, including the stop bit, are transmitted in the Resource Data (RDATA) field. This history information can be concatenated by the development tool with the branch/predicate history information from subsequent messages to obtain the complete branch/predicate history between indirect changes of flow.

For instruction counter overflow, the Resource Full Message transmits an RCODE of 0b0000 and a value of 0xFF is transmitted in the RDATA field, indicating that 255 sequential instructions have been executed since the last change of flow, or, if program trace is in history mode, since the last instruction that recorded history information.

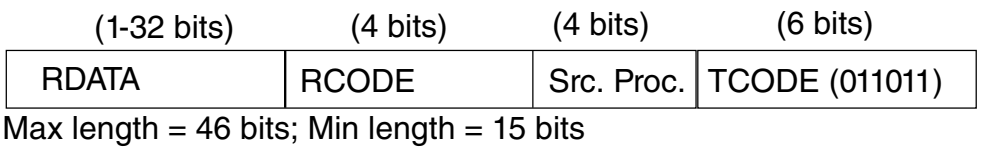


Figure 67-37. Resource Full Message format

The following table shows the RCODE encodings and RDATA information used for Resource Full Messages.

Table 67-32. RCODE encoding

RCODE	Description	RDATA field
0000	Program Trace Instruction counter reached 255 and was reset.	0xFF
0001	Program Trace, Branch / Predicate Instruction History full.	Branch HIstory. This type of packet is terminated by a stop bit set to 1 after the last history bit.

67.12.5 Program Correlation Messages

Program Correlation Messages (PCMs) are used to correlate events to the program flow that may or may not be associated with the instruction stream. The following events will result in a PCM when program trace is enabled:

- When the CPU enters debug mode, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to debug mode entry.
- When the CPU first enters a low power mode in which instructions are no longer executed, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to low power mode entry.
- Whenever program trace is disabled by any means, a PCM is generated. The instruction count and history information provided by the PCM can be used to determine the last sequence of instructions executed prior to disabling program trace.
- When a "Branch and Link" instruction executes (direct branch function call — **bl/bcl/bla/bcla**-type instructions)
- When program trace becomes masked due to MSR_{PMM}='0' and $DC4_{PTMARK}$='1'.
- When a write to the process ID register selected by $DC1_{OTS}$ (PID0 or NPIDR) is made via a **mtspr** PID0 or **mtspr** NPIDR.

Refer to Table 67-6 for the event codes that are supported in this implementation. Event code masking is available via the EVCDM field of the DC4 register to allow for control over generation of Program Correlation messages for each event type.

Program Correlation is messaged out in the following formats:

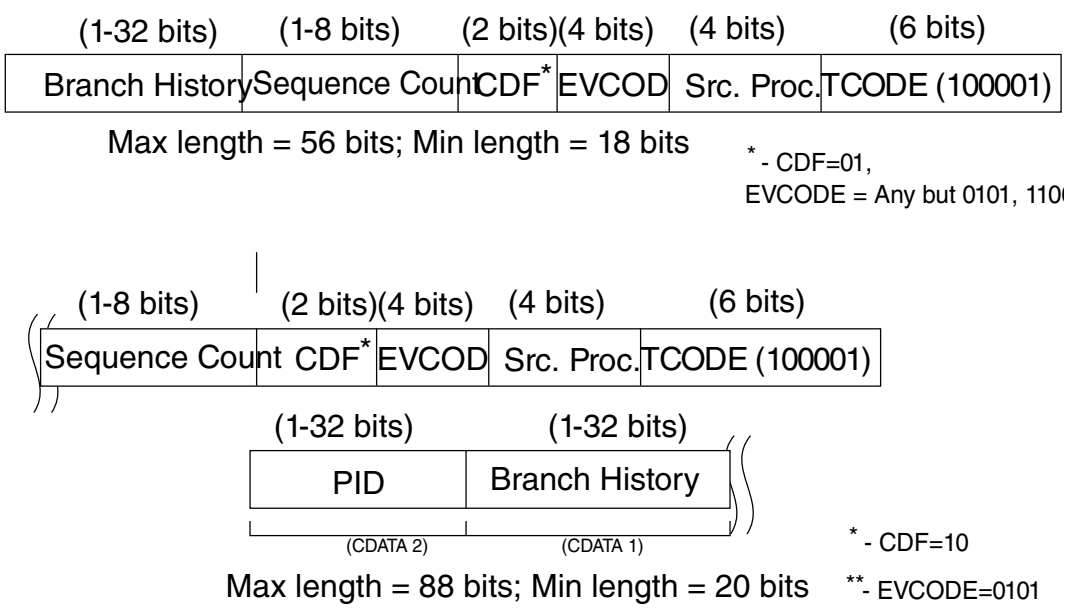


Figure 67-38. Program Correlation message formats

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

67.12.5.1 Program Correlation message generation for PID updates

When a (potentially) new value is established in the selected process ID register via a **mtspr** PID0/NPIDR, a Program Correlation message is generated containing the information regarding the new process ID value. This PCM also contains the current history and instruction count. The message is provided so that address translation information can be processed by the development tool in the proper program flow. The **mtspr** PID0/NPIDR is included in the instruction count information. Note that Ownership Trace Messages (other than the periodic OTM) are redundant with the information provided, and may be disabled to avoid unnecessary message bandwidth or collisions.

67.12.6 Program Trace Overflow Error messages

An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard incoming messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied.

67.12.7 Program Trace Synchronization messages

By default, program trace messages will perform XOR compression on the branch target address to produce the address field for the message. This compression is consistent with the specification in IEEE-ISTO 5001.

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 67-33):

- Initial Program Trace message after exit from system reset or whenever program trace is enabled.
- Upon exiting from a CPU Low Power state.
- Upon exiting from Debug Mode.

- Upon occurrence of queue overrun (can be caused by any trace message), provided Program Trace is enabled.
- Upon assertion of the Event In (**nex_evti_b**) pin if the EIC bits within the DC1 Register have enabled this feature.
- When program trace becomes unmasked due to $MSR_{PMM} \rightarrow$ '1' with $DC4_{PTMARK}$='1'.

Note that the ICNT information for these messages is driven to zero, thus will not always be meaningful for some of these cases.

The format for Program Trace Synchronization messages is as follows:

(1-32 bits)	(1-8 bits)	(1 bit)	(4 bits)	(6 bits)
Full Target Address	Seq. Count (0)	(0)	Source Proc.	TCODE (001001)

Max length = 51 bits; Min length = 13 bits

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

Exception conditions that result in Program Trace Synchronization are summarized in Table 67-33.

Table 67-33. Program Trace exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. Upon exiting system reset (**p_reset_b**), if Program Trace is already enabled, a Program Trace Sync Message is sent.
Program Trace Enabled	The first Program Trace Message (after Program Trace has been enabled or re-enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode, a Program Trace Sync Message is sent.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next BTM message in the queue will be a Program Trace Sync Message.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Program Trace Sync Message if Program Trace is enabled and the EIC bits of the DC1 Register have enabled this feature.

Note that for cases where program trace sync messages are generated due to program trace being enabled, the address contained in the sync message may either be the address of the instruction that caused program trace to be enabled, or may be the address of the first instruction of an exception handler that is executed in response to unsuccessful completion of that instruction.

67.12.8 Program Trace Direct/Indirect Branch with Sync messages

Under some conditions an uncompressed address is sent to provide development tools with a baseline reference address. A Program Trace Synchronization or a Direct/Indirect Branch with Sync message is messaged via the auxiliary port (provided Program Trace is enabled) for the following conditions (see Table 67-33):

- Upon direct/indirect branch after a Program Sync Message was lost due to a collision while attempting to enter the message queue.
- Upon direct/indirect branch after the sequential instruction counter has expired indicating 255 instructions have occurred since the last change of flow.
- When the periodic program trace counter has expired indicating 255 *without-sync* Program Trace messages have occurred since the last *with-sync* message occurred.

Note that the ICNT and History information for the first message will not always be meaningful, since the temporary masking of program trace may result in ambiguous values. Subsequent w/sync messages will not have this issue.

The format for Program Trace Direct/Indirect Branch with Sync Messages is as follows:

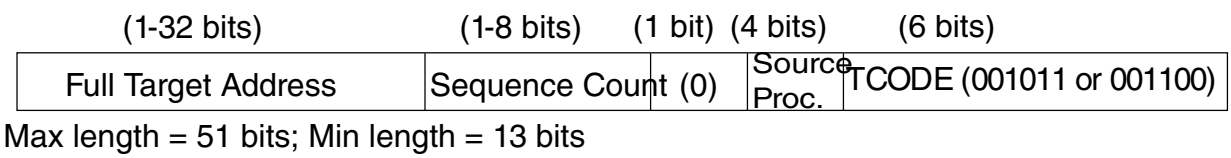


Figure 67-39. Direct/Indirect Branch with Sync message format

The format for Program Trace Indirect Branch History with Sync. messages is as follows:

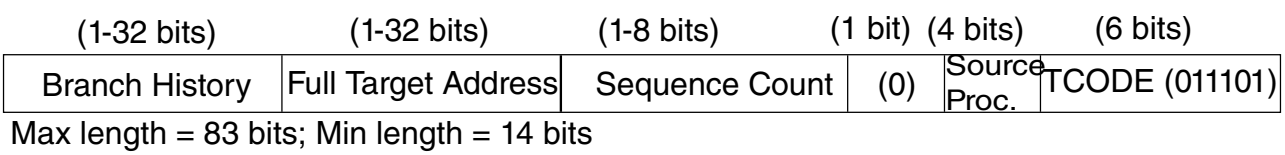


Figure 67-40. Indirect Branch History w/ Sync. message format

Exception conditions that result in Program Trace Synchronization using a w/Sync message type are summarized in Table 67-34.

Table 67-34. Program Trace Exception Summary for w/Sync BTM messages

Exception Condition	Exception Handling
Periodic Program Trace Sync.	A forced synchronization occurs periodically after 255 non-sync Program Trace Messages have been queued. A Direct/Indirect Branch w/ Sync. Message is queued. The periodic program trace message counter then resets.
Sequential Instruction Count Overflow	After the sequential instruction counter reaches its maximum count (up to 255 sequential instructions may be executed), a forced synchronization occurs. The sequential counter then resets. A Program Trace Direct/Indirect Branch w/ Sync.Message is queued upon execution of the next branch. A Resource Full Message is Queued on the overflow event. If a branch instruction is the 255th instruction to occur, and causes a Program Trace message to be queued, then no Resource Full Message is queued, and the w/Sync message will be queued for the *next* Program Trace Direct/Indirect Branch Message.
Collision Priority	All Messages have the following priority: Instruction 0 (WPM -> DQM -> PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A BTM Message from Instruction1 that attempts to enter the queue at the same time as three higher priority messages from either instruction will be lost. An Error Message will be sent indicating the BTM was lost. The following direct/indirect branch will queue a Direct/Indirect Branch w/ Sync. Message. The count value within this message will reflect the number of sequential instructions executed after the last successful BTM Message was generated. This count will include the branch that did not generate a message due to the collision.

67.12.9 Enabling Program Trace

Program Trace Messaging can be enabled in one of three ways:

- Setting the TM field of the DC1 Register to enable Program Trace
- Using the PTS field of the WT Register to enable Program Trace on Watchpoint hits (watchpoints are configured within the CPU)
- Filtering of Program Trace messages may be performed using the MSR_{PMM} bit and the setting of $DC4_{PTMARK}$

67.12.10 Program Trace Timing Diagrams (2 MDO / 1 MSEO configuration)

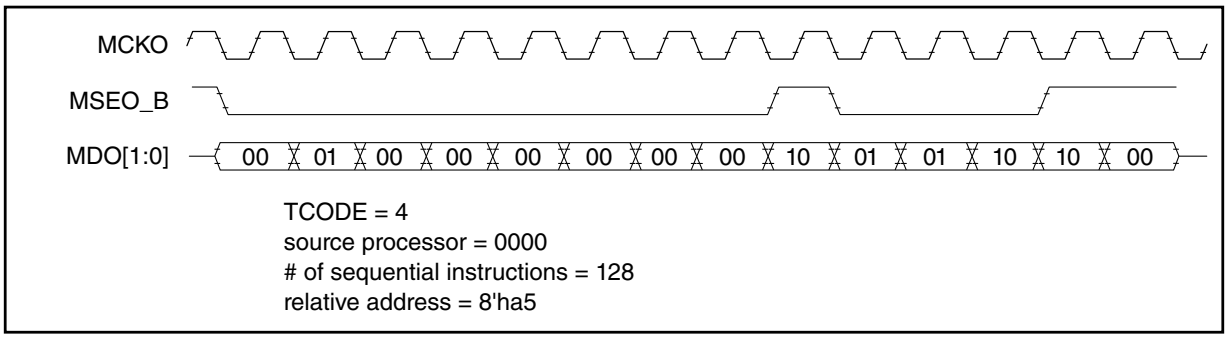


Figure 67-41. Program Trace - Indirect Branch message (Traditional)

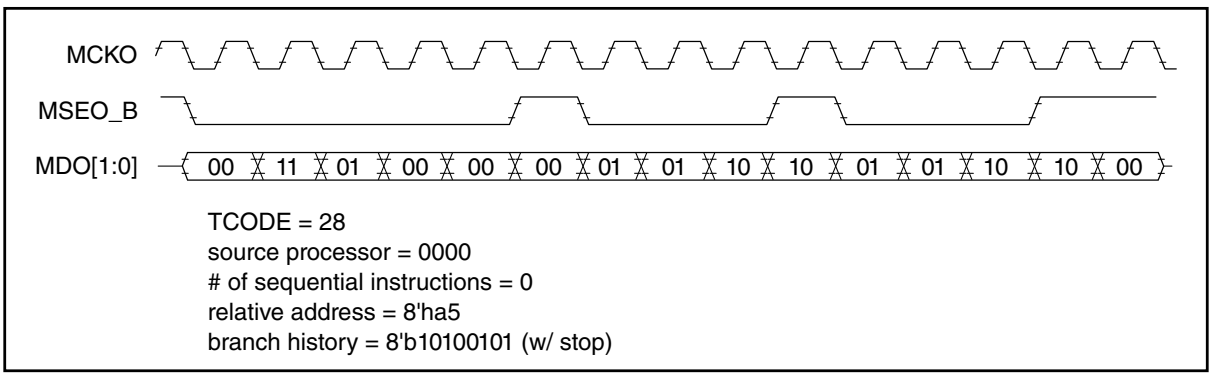


Figure 67-42. Program Trace - Indirect Branch message (history)

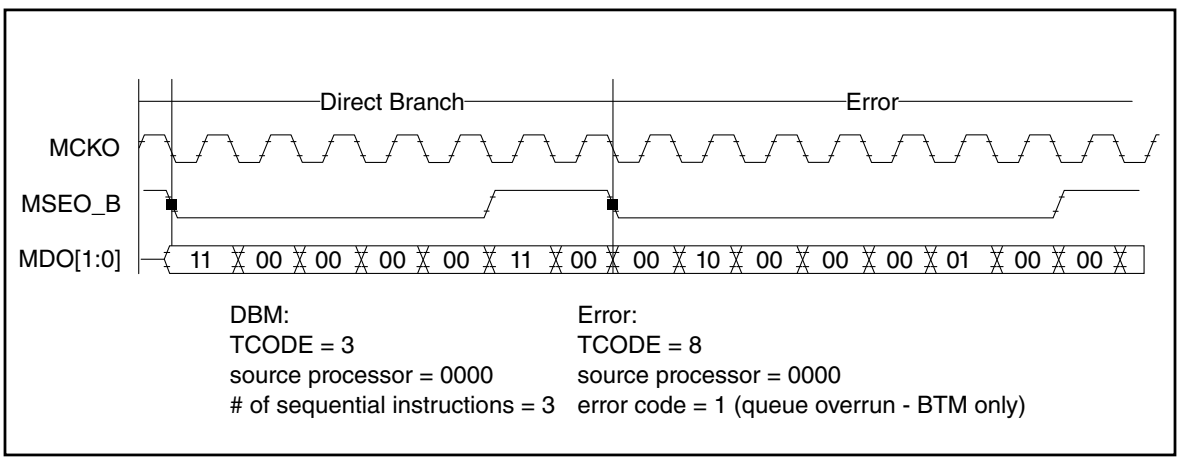


Figure 67-43. Program Trace - Direct Branch (traditional) & Error messages

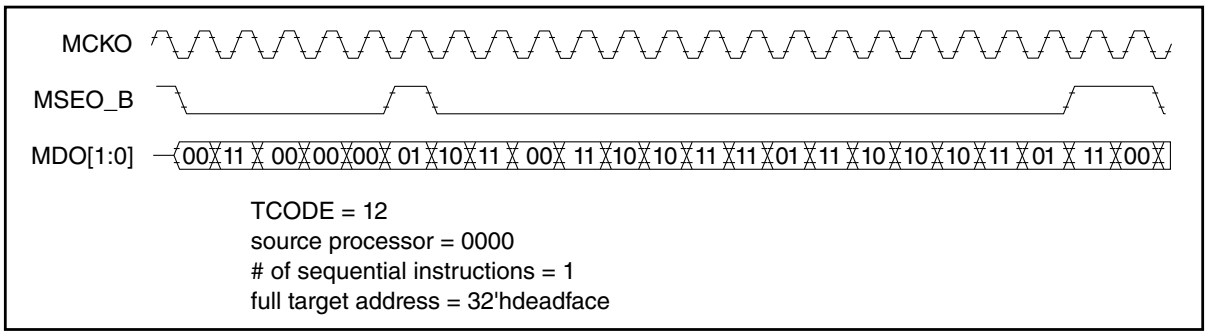


Figure 67-44. Program Trace - Indirect Branch w/ Sync. message

67.13 Data Trace

This section deals with the Data Trace mechanism supported by the Nexus 3 module. Data Trace is implemented via Data Write Messaging (DWM) and Data Read Messaging (DRM), as per the **IEEE-ISTO 5001** standard.

67.13.1 Data Trace Messaging (DTM)

Data Trace Messaging is accomplished by snooping the internal address and data buses, and storing the information for qualifying accesses (based on enabled features and matching target addresses). The Nexus 3 module traces all data access that meet the selected range and attributes.

Note

Data Trace is only performed on the internal data buses. This allows for data visibility for Z4 Core processors that incorporate a data cache. Only CPU initiated accesses will be traced. No DMA accesses to the AHB system bus will be traced.

Data Trace Messaging can be enabled in one of three ways.

- Setting the TM field of the DC1 Register to enable Data Trace.
- Using the DTS field of the WT Register to enable Data Trace on Watchpoint hits (watchpoints are configured within the Nexus1 module).

67.13.2 DTM Message formats

The Nexus 3 block supports five types of DTM Messages — Data Write, Data Read, Data Write Synchronization, Data Read Synchronization and Error Messages.

67.13.2.1 Data Write messages

The Data Write message contains the data write value and the address of the write access, relative to the previous Data Trace Message. Data Write message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000101)

Max length = 111 bits; Min length = 17 bits

Figure 67-45. Data Write message format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

67.13.2.2 Data Read messages

The Data Read message contains the data read value and the address of the read access, relative to the previous Data Trace message. Data Read message information is messaged out in the following format:

(1-64 bits)	(1-32 bits)	(4 bits)	(1 bit)	(4 bits)	(6 bits)
Data Value(s)*	Relative Address	Data Size	(0)	Src. Proc	TCODE (000110)

Max length = 111 bits; Min length = 17 bits

Figure 67-46. Data Read fessage format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

Note

Z4 based CPUs are capable of generating two (2) reads or writes per clock cycle in cases where multiple registers are accessed with a single instruction (lmw/stmw). These will have a double word pair size encoding (**p_tsiz** = 0b000). In these

cases, the nexus3 module will send one (1) Data Trace Message with the two 32-bit data values as one combined 64-bit value for each message.

For Z4 based CPUs, the doubleword encoding (**p_tsiz** = 0b000) may also indicate a doubleword access and will be sent out as a single Data Trace Message with a single 64-bit data value.

The debug/development tool will need to distinguish the two cases based on the family of processor.

67.13.2.3 Data Trace Synchronization messages

A Data Trace Write/Read with Sync. message is messaged via the auxiliary port (provided Data Trace is enabled) for the following conditions (see Table 67-35):

- Initial Data Trace Message after exit from system reset or whenever Data Trace is enabled.
- Upon returning from a CPU Low Power state.
- Upon returning from Debug Mode.
- After occurrence of queue overrun (can be caused by any trace message), provided Data Trace is enabled.
- After the periodic data trace counter has expired indicating 255 *without-sync* Data Trace messages have occurred since the last *with-sync* message occurred.
- Upon assertion of the Event In (**nex_evti_b**) pin, the first Data Trace Message will be a synchronization message if the EIC bits of the DC1 Register have enabled this feature.
- Upon Data Trace Write/Read after the previous DTM message was lost due to an attempted access to a secure memory location (for SOC's w/ security).
- Upon Data Trace Write/Read after the previous DTM message was lost due to a collision entering the FIFO between the DTM message and any two of the following: Watchpoint message, Ownership Trace message, or Program Trace message.

Data Trace Synchronization Messages provide the full address (without leading zeros) and insure that development tools fully synchronize with Data Trace regularly. Synchronization messages provide a reference address for subsequent DTMs, in which only the unique portion of the Data Trace address is transmitted. The format for Data Trace Write/Read with Sync. Messages is as follows:

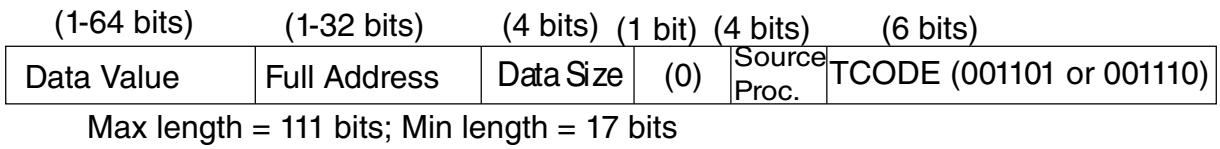


Figure 67-47. Data Write/Read with Sync. message format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

Exception conditions that result in Data Trace Synchronization are summarized in Table 67-35.

Table 67-35. Data Trace Exception summary

Exception Condition	Exception Handling
System Reset Negation	At the negation of JTAG reset (**j_trst_b**), queue pointers, counters, state machines, and registers within the Nexus 3 module are reset. If Data Trace is enabled, the first Data Trace Message is a Data Write/Read w/ Sync. Message.
Data Trace Enabled	The first Data Trace Message (after Data Trace has been enabled) is a synchronization message.
Exit from Low Power/Debug	Upon exit from a Low Power mode or Debug mode the next Data Trace Message will be converted to a Data Write/Read with Sync. Message.
Queue Overrun	An Error Message occurs when a new message cannot be queued due to the message queue being full. The FIFO will discard messages until it has completely emptied the queue. Once emptied, an Error Message will be queued. The error encoding will indicate which type(s) of messages attempted to be queued while the FIFO was being emptied. The next DTM message in the queue will be a Data Write/Read w/ Sync. Message.
Periodic Data Trace Sync.	A forced synchronization occurs periodically after 255 Data Trace Messages have been queued. A Data Write/Read w/ Sync. Message is queued. The periodic data trace message counter then resets.
Event In	If the Nexus module is enabled, a **nex_evti_b** assertion initiates a Data Trace Write/Read w/ Sync. Message upon the next data write/read (if Data Trace is enabled and the EIC bits of the DC1 Register have enabled this feature).
Collision Priority	All Messages have the following priority: Instruction 0 (WPM → DQM → PCM_{PIDMSG} → OTM → BTM → DTM) → Instruction1 (WPM → DQM → OTM → BTM → DTM), where instruction0 is the oldest instruction. A DTM Message that attempts to enter the queue at the same time as three other higher priority messages will be lost. A subsequent read/write will queue a Data Trace Read/Write w/ Sync. Message.

67.13.3 DTM Operation

67.13.3.1 Data Trace windowing

Data Write/Read Messages are enabled via the RWT field in the Data Trace Control Register (DTC) for each DTM channel. Data Trace windowing is achieved via the address range defined by the DTEA and DTSA Registers and by the RC field in the DTC register. All CPU initiated read/write accesses that fall inside or outside these address ranges, as programmed, are candidates to be traced.

67.13.3.2 Data Access / Instruction Access Data Tracing

The Nexus3 module is capable of tracing either instruction access data or data access data and can be configured for either type of data trace by setting the DI1 field within the Data Trace Control Register. This setting applies to all DTM channels.

67.13.3.3 Data Trace filtering

Data Trace filtering is available base on the settings of MSR_{PMM} and $DC4_{DTMARK}$.

67.13.3.4 Bus Cycle special cases

Table 67-36. Bus Cycle Special Cases

Special Case	Action
Bus cycle aborted	Cycle ignored
Bus cycle with data error ($\overline{TEA}$), [1]	Data Trace Message discarded
Bus cycle completed without error[1]	Cycle captured & transmitted
AHB bus cycle initiated by Nexus 3	Cycle ignored
Bus cycle is an instruction fetch	Cycle selectively ignored based on DTC_{DI} setting
Bus cycle accesses misaligned data (across 64-bit boundary) - both 1st & 2nd transactions within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction out of data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction within data trace range; 2nd transaction (regardless of within range or not) receives a bus error	Data Trace Message discarded
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within data trace range	1st & 2nd cycle captured & a single or a pair of DTM(s) is (are) transmitted (see Note)

Table continues on the next page...

Table 67-36. Bus Cycle Special Cases (continued)

Special Case	Action
Bus cycle accesses misaligned data (across 64-bit boundary) - 1st transaction out of data trace range; 2nd transaction within range, receives a bus error	Data Trace Message discarded

1. Buffering of stores in the CPU store buffer may generate a DTM prior to the actual memory access, regardless of an error termination condition from memory.

Note

For misaligned accesses (crossing 64-bit boundary), the access is broken into two accesses by the CPU. If either access is within the data trace range, a single DTM will be sent with a size encoding indicating the size of the original access (i.e. word), and the address indicating the original misaligned accesses, unless the misaligned access wraps into the doubleword at address 0. In this case, since the two portions of the misaligned access are not contiguous, two DTMs will be sent, one for each portion. The size encodings and the addresses of the DTMs will indicate the accessed bytes of data. The data trace port handles these cases in the same manner. (See the Data Trace Port section in the Z4 Core Debug Support chapter.)

Note

A store to the cache's store buffer within the data trace range may initiate a DTM message prior to completion of the actual memory access.

67.13.4 Data Trace Timing Diagrams (8 MDO / 2 MSEO configuration)

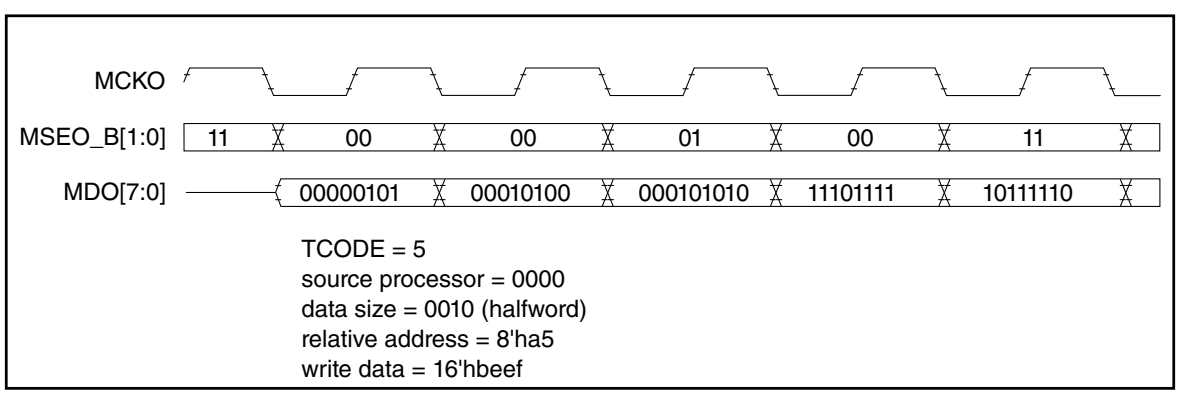


Figure 67-48. Data Trace - Data Write Message

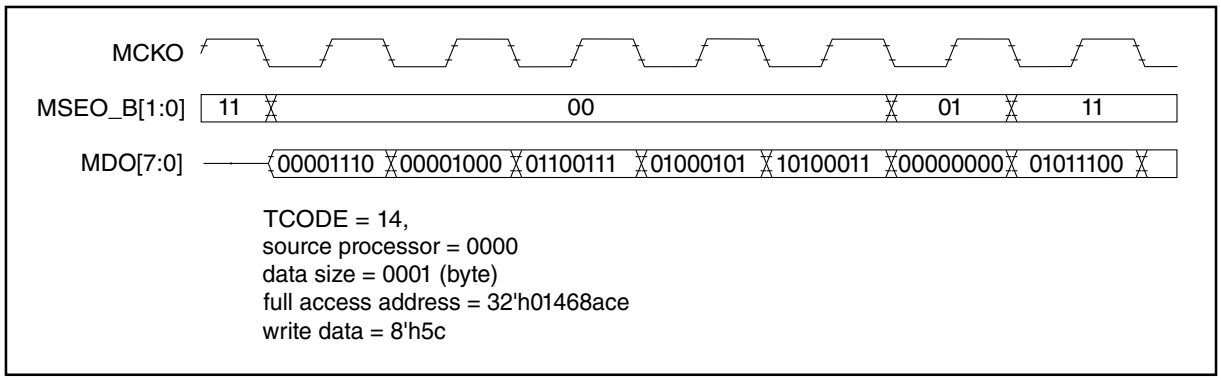


Figure 67-49. Data Trace — Data Read w/ Sync message

67.14 Data Acquisition messaging

This section details the Data Acquisition mechanisms supported by the Nexus 3 module. Data Acquisition Trace is implemented using Data Acquisition Trace Messages in accordance with IEEE-ISTO 5001 definitions. The control mechanism to export the data is different from the recommendations of the standard, however.

Data Acquisition Trace provides a convenient and flexible mechanism for the debugger to observe the architectural state of the machine through software instrumentation.

67.14.1 Data Acquisition ID Tag field

The DQTAG Tag field (DQTAG) is an 8-bit value specifying control or attribute information for the data included in the Data Acquisition message. DQTAG is sampled from $DEVENT_{DQTAG}$ when a write to DDAM is performed via **mtspr** operations. The usage of the DQTAG is left to the discretion of the development tool to be used in whatever manner is deemed appropriate for the application.

67.14.2 Data Acquisition Data field

The Data Acquisition Data field (DQDATA) is the data captured from the DDAM write operation via **mtspr** operations. Leading zeros are omitted from the message.

67.14.3 Data Acquisition Trace event

For DQM, a dedicated SPR has been allocated (DDAM). It is expected that the general use case is to instrument the software and use **mtspr** operations to generate Data Acquisition messages.

There is no explicit error response for failed accesses as a result of contention between an internal and external debugger. Software may be blocked or given ownership of DDAM and the DQTAG field of the DEVENT register via control in EDBRAC0 while in External Debug Mode. Hardware always has access to these registers. Refer to the "External Debug Resource Allocation Control Register (EDBRAC0)" section in the Z4 Core Debug Support chapter for more detail on EDBRAC0.

Reads from the Data Acquisition channel do not generate a Data Acquisition event and will return zeroes for the read data.

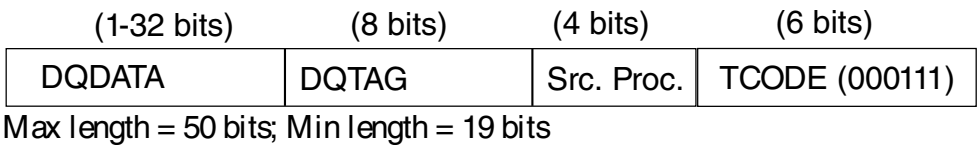


Figure 67-50. Data Acquisition message format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

67.15 Watchpoint Trace messaging

Enabling Watchpoint messaging is done by setting the Watchpoint Trace Enable bit in the DC1 Register. Setting the individual Watchpoint sources is supported through the Nexus1 module and the Performance Monitor unit. The Nexus1 module is capable of setting multiple types of watchpoints. Please refer to the Z4 Core Core Debug Support chapter for details on watchpoint initialization.

When watchpoints occur due to one or more asserted watchpoint event signals and Watchpoint Trace Messaging is enabled, a Watchpoint Trace message will be sent to the message queue to be messaged out. This message includes the watchpoint number indicating which watchpoint(s) caused the message. If more than one enabled watchpoint occurs in a single cycle, only one Watchpoint Trace message is generated and multiple bits of the watchpoint hit field will be set. The settings of the $WMSK_{WEM}$ field control which watchpoints are enabled to generate watchpoint trace messages.

The occurrence of any of the defined watchpoints can also be programmed to assert the Event Out (**nex_evto_b**) pin for one (1) period of the output clock (**nex_mcko**) based on settings in the DC2 and DC3 registers. See Table 67-39 for details on **nex_evto_b**.

Watchpoint information is messaged out in the following format:

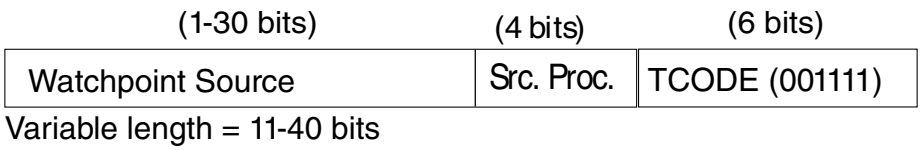


Figure 67-51. Watchpoint message format

If timestamping is enabled, the TSTAMP field is appended to the end of the message and adds up to 30 additional bits.

The Watchpoint Source message field will contain a '1' for each asserted watchpoint. Leading zeros are truncated.

Table 67-37. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
000000000000000000000000000000	No Watchpoints enabled for Watchpoint Trace Messaging
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX1	Watchpoint #0 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXXXXXXXX1X	Watchpoint #1 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXXXXXXX1XX	Watchpoint #2 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXXXXXX1XXX	Watchpoint #3 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXXXXX1XXXX	Watchpoint #4 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXXXX1XXXXX	Watchpoint #5 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXXX1XXXXXX	Watchpoint #6 enabled for WTM
XXXXXXXXXXXXXXXXXXXXXX1XXXXXXX	Watchpoint #7 enabled for WTM
XXXXXXXXXXXXXXXXXXXXX1XXXXXXXX	Watchpoint #8 enabled for WTM
XXXXXXXXXXXXXXXXXXXX1XXXXXXXXX	Watchpoint #9 enabled for WTM
XXXXXXXXXXXXXXXXXXX1XXXXXXXXXX	Watchpoint #10 enabled for WTM
XXXXXXXXXXXXXXXXXX1XXXXXXXXXXX	Watchpoint #11 enabled for WTM
XXXXXXXXXXXXXXXXX1XXXXXXXXXXXX	Watchpoint #12 enabled for WTM
XXXXXXXXXXXXXXXX1XXXXXXXXXXXXX	Watchpoint #13 enabled for WTM
XXXXXXXXXXXXXXX1XXXXXXXXXXXXXX	Watchpoint #14 enabled for WTM
XXXXXXXXXXXXXX1XXXXXXXXXXXXXXX	Watchpoint #15 enabled for WTM
XXXXXXXXXXXXX1XXXXXXXXXXXXXXXX	Watchpoint #16 enabled for WTM
XXXXXXXXXXXX1XXXXXXXXXXXXXXXXX	Watchpoint #17 enabled for WTM
XXXXXXXXXXX1XXXXXXXXXXXXXXXXXX	Watchpoint #18 enabled for WTM
XXXXXXXXXX1XXXXXXXXXXXXXXXXXXX	Watchpoint #19 enabled for WTM
XXXXXXXXX1XXXXXXXXXXXXXXXXXXXX	Watchpoint #20 enabled for WTM
XXXXXXXX1XXXXXXXXXXXXXXXXXXXXX	Watchpoint #21 enabled for WTM

Table 67-37. Watchpoint Source Encoding

Watchpoint Source (1-30 bits)	Watchpoint Description
XXXXXXX1XXXXXXXXXXXXXXXXXXXXXX	- Watchpoint #22 enabled for WTM
XXXXXX1XXXXXXXXXXXXXXXXXXXXXXX	- Watchpoint #23 enabled for WTM
XXXXX1XXXXXXXXXXXXXXXXXXXXXXXX	- Watchpoint #24 enabled for WTM
XXXX1XXXXXXXXXXXXXXXXXXXXXXXXX	- Watchpoint #25 enabled for WTM
XXX1XXXXXXXXXXXXXXXXXXXXXXXXXX	- Watchpoint #26 enabled for WTM
XX1XXXXXXXXXXXXXXXXXXXXXXXXXXX	- Watchpoint #27 enabled for WTM
X1XXXXXXXXXXXXXXXXXXXXXXXXXXXX	- Watchpoint #28 enabled for WTM
1XXXXXXXXXXXXXXXXXXXXXXXXXXXXX	- Watchpoint #29 enabled for WTM

67.15.1 Watchpoint Timing Diagram (2 MDO / 1 MSEO configuration)

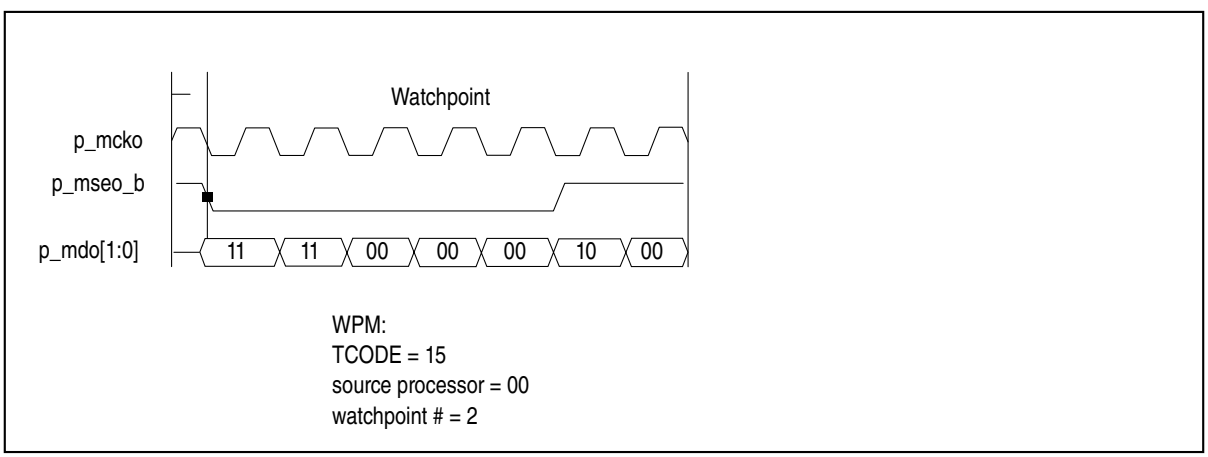


Figure 67-52. Watchpoint Message & Watchpoint Error message

67.16 Nexus 3 Read/Write access to memory-mapped resources

The Read/Write access feature allows access to memory-mapped resources via the JTAG/ OnCE port. The Read/Write mechanism supports single as well as block reads and writes to AHB resources.

The Nexus 3 module is capable of accessing resources on the system bus (AHB). Memory-mapped registers and other non-cached memory can be accessed via the standard memory map settings.

All accesses are setup and initiated by the Read/Write Access Control/Status Register (RWCS), as well as the Read/Write Access Address (RWA) and Read/Write Access Data Registers (RWD). Nexus 3 read/write accesses are run as privileged data non-cacheable accesses by default, and drive the **p_d_hprot[5:0]** bus access attributes to 6'b000011 accordingly. The $RWCS_{ATTR}$ field is provided to allow a portion of these default values to be modified when performing read or write accesses using the Nexus 3 Read/Write access mechanism.

Using the Read/Write Access Registers (RWCS/RWA/RWD), memory mapped AHB resources can be accessed through Nexus 3. The following subsections describe the steps required to access memory-mapped resources.

Note

Read/Write Access can only access memory mapped resources when system reset is de-asserted and clocks are running.

Misaligned accesses are NOT supported in the Nexus 3 module.

Uncorrectable ECC errors on Nexus 3 read access will result in the RWD register being updated with the raw data received.

67.16.1 Single write access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Address → 32h'xxxxxxxx (write address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)
   - Access Priority (PR) → 2b'11 (highest priority)
   - Read/Write (RW) → 1b'1 (write access)
   - Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note

Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. Initialize the Read/Write Access Data Register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Write Data → 32h'xxxxxxxx (write data)
4. The Nexus block will then arbitrate for the AHB system bus and transfer the data value from the data buffer RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed, Nexus clears the AC and DV bits in the RWCS Register. If the access has completed without error, (ERR=1'b0), Nexus asserts the **nex_rdy_b** pin (see Table 67-39 for details) and clears the ERR bit in the RWCS Register. Otherwise, if the access completes with an error, the **nex_err_b** pin will be asserted and the ERR bit will be set to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once DV has been cleared, this indicates that the device is ready for the next access, or that an error has occurred.

Note

Only the **nex_ahb_start, nex_rdy_b,** and **nex_err_b** pins as well as the AC, DV, and ERR bits within the RWCS provide Read/Write Access status to the external development tool.

67.16.2 Block write access

1. For a block write access, follow Steps 1, 2, and 3 outlined in Single write access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and transfer the first data value from the RWD Register to the memory mapped address in the Read/Write Access Address Register (RWA). The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, and

the **nex_rdy_b** pin is asserted. If the access has completed without error, Nexus clears the ERR and DV bits in the RWCS Register, otherwise the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the **nex_err_b** pin is asserted, the block transfer is aborted, and the AC bit in the RWCS register is cleared. Once DV has been cleared, this indicates that the device is ready for the next access in the block transfer, or an error has occurred.

3. If AC has not been cleared due to an error, repeat Step 3 in Single write access until the internal CNT value is zero (0). When this occurs, the AC bit within the RWCS will be cleared to indicate the end of the block write access.

Note

The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block write access. The original values can be read by the external development tool at any time.

67.16.3 Single read access

1. Initialize the Read/Write Access Address Register (RWA) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure as follows:
   - Read Address → 32h'xxxxxxxx (read address)
2. Initialize the Read/Write Access Control/Status Register (RWCS) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE. Configure the bits as follows:
   - Access Control (AC) → 1b'1 (to indicate start access)
   - Map Select (MAP) → 3b'000 (primary memory map)
   - Access Priority (PR) → 2b'11 (highest priority)
   - Read/Write (RW) → 1b'0 (read access)
   - Word Size (SZ) → 3b'0xx (32-bit, 16-bit, 8-bit)
   - Access Count (CNT) → 14h'0000 or 14h'0001(single access)

Note

Access Count (CNT) of 14'h0000 or 14'h0001 will perform a single access.

3. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and asserts the **nex_rdy_b** pin (see Table 67-39 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.

Note

Only the **nex_ahb_start, nex_rdy_b,** and **nex_err_b** pins as well as the AC, DV and ERR bits within the RWCS provide Read/Write Access status to the external development tool.

67.16.4 Block read access

1. For a block read access, follow Steps 1 and 2 outlined in Single read access to initialize the registers, but using a value greater than one (14'h0001) for the CNT field in the RWCS Register.
2. The Nexus block will then arbitrate for the AHB system bus and the read data will be transferred from the AHB to the RWD Register. The nex_ahb_start output will be asserted during the first clock of the address phase of the transfer. When the access has completed without error, Nexus clears the ERR bit and sets the DV bit in the RWCS Register and asserts the **nex_rdy_b** pin (see Table 67-39 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. The AC bit will be cleared in either case.
3. When the transfer has completed without error, the address from the RWA Register is incremented to the next word size (specified in the SZ field), the number from the CNT field is decremented, Nexus clears the ERR bit and sets the DV bit in the

RWCS Register, and asserts the **nex_rdy_b** pin (see Table 67-39 for detail on **nex_rdy_b**). Otherwise, if the access has completed with an error, the **nex_err_b** pin will be asserted, the ERR bit will be set and the DV bit will be cleared to indicate an error has occurred, the AC bit is cleared and the block transfer is aborted, and the **nex_rdy_b** pin will not be asserted. Once ERR or DV has been set, this indicates that the device is ready for the next access, or that an error has occurred. Once DV has been set, this indicates that the device is ready for the next access in the block transfer, or if ERR is set (AC will be cleared), the block transfer has been aborted.

4. The data can then be read from the Read/Write Access Data Register (RWD) through the access method outlined in Nexus 3 Register Access via JTAG/OnCE.
5. If AC has not been cleared due to an error, repeat Steps 3 and 4 in Single read access until the CNT value is zero (0). When this occurs, the AC bit within the RWCS is cleared to indicate the end of the block read access.

Note

The data values must be shifted out 32 bits at a time LSB first (i.e. doubleword read = two word reads from the RWD).

Note

The actual RWA value as well as the CNT field within the RWCS are not changed when executing a block read access. The original values can be read by the external development tool at any time.

67.16.5 Error handling

The Nexus 3 module handles various error conditions as follows:

67.16.5.1 AHB Read/Write error

All address and data errors that occur on read/write accesses to the AHB system bus will return a transfer error encoding on the **p_hresp[1:0]** signals. If this occurs:

1. The access is terminated without retrying (AC bit is cleared)
2. The ERR bit in the RWCS Register is set

3. The Error Message is sent (TCODE = 8) indicating Read/Write error

67.16.5.2 Access termination

The following cases are defined for sequences of the Read/Write protocol that differ from those described in the above sections.

1. If the AC bit in the RWCS Register is set to start Read/Write accesses and invalid values are loaded into the RWD and/or RWA, then an AHB access error may occur. This is handled as described above.
2. If a block access is in progress (all cycles not completed), and the RWCS Register is written, then the original block access is terminated at the boundary of the nearest completed access.
   a. If the RWCS is written with the AC bit set, the next Read/Write access will begin and the RWD can be written to/ read from.
   b. If the RWCS is written with the AC bit cleared, the Read/Write access is terminated at the nearest completed access. This method can be used to break (early terminate) block accesses.

67.16.6 Read/Write access error message

The Read/Write Access Error Message is sent out when an AHB system bus access error (read or write) has occurred.

Error information is messaged out in the following format:

(5 bits)	(4 bits)	(6 bits)
Error Code (00011)	Src. Proc.	TCODE (001000)

Fixed length = 15 bits

Figure 67-53. Error message format1

67.17 Nexus 3 pin interface

This section details information regarding the Nexus 3 pins and pin protocol.

The Nexus 3 pin interface provides the function of transmitting messages from the messages queues to the external tools. It is also responsible for handshaking with the message queues.

67.17.1 Pins implemented

The Nexus 3 module implements an auxiliary port consisting of one (1) **nex_evti_b** and one (1) **nex_mseo_b** or two (2) **nex_mseo_b[1:0]**. It also implements a configurable number of **nex_mdo[n:0]** pins, (1) **nex_err_b** pin, (1) **nex_evto_b** pin, and one (1) clock output pin (**nex_mcko**), as well as additional configuration pins described in Table 67-39. The output pins are synchronized to the Nexus 3 output clock (**nex_mcko**).

All Nexus 3 input functionality may be controlled through the JTAG/OnCE port in compliance with IEEE 1149.1 (see Nexus 3 Register Access via JTAG/OnCE for details). The JTAG pins are incorporated as I/O to the processor, and are further described in the "JTAG/OnCE Pins" section of the Z4 Core Core Debug Support chapter.

Table 67-38. JTAG Pins for Nexus 3

JTAG Pins	Input/ Output	Description of JTAG Pins (included in Nexus 1)
j_tdo	O	The Test Data Output (**j_tdo**) pin is the serial output for test instructions and data. **j_tdo** is three-stateable and is actively driven in the "Shift-IR" and "Shift-DR" controller states. **j_tdo** changes on the falling edge of **j_tclk**.
j_tdi	I	The Test Data Input (**j_tdi**) pin receives serial test instruction and data. TDI is sampled on the rising edge of **j_tclk**.
j_tms	I	The Test Mode Select (**j_tms**) input pin is used to sequence the OnCE controller state machine. **j_tms** is sampled on the rising edge of **j_tclk**.
j_tclk	I	The Test Clock (**j_tclk**) input pin is used to synchronize the test logic, and control register access through the JTAG/OnCE port.
j_trst_b	I	The Test Reset (**j_trst_b**) input pin is used to asynchronously initialize the JTAG/OnCE controller.

The auxiliary pins are used to send and receive messages and are described in Table 67-39.

Table 67-39. Nexus 3 Auxiliary pins

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mcko	O	Message Clock Out (**nex_mcko**) is a free running output clock to development tools for timing of **nex_mdo[n:0]** & **nex_mseo_b[1:0]** pin functions. **nex_mcko** is programmable through the DC1 Register.
nex_mdo[n:0]	O	Message Data Out (**nex_mdo[n:0]**) are output pins used for OTM, BTM, and DTM. External latching of **nex_mdo[n:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**).

Table continues on the next page...

Table 67-39. Nexus 3 Auxiliary pins (continued)

Auxiliary pins	Input/ Output	Description of auxiliary pins
nex_mseo_b[1:0]	O	Message Start/End Out (**nex_mseo_b[1:0]**) are output pins that indicate when a message on the **nex_mdo[n:0]** pins has started, when a variable length packet has ended, and when the message has ended. External latching of **nex_mseo_b[1:0]** shall occur on the rising edge of the Nexus3 clock (**nex_mcko**). One or two pin MSEO functionality is determined at integration time per SOC implementation.
nex_err_b	O	Error (**nex_err_b**) is an output pin used to indicate to the external tool that the Nexus block is ready for the next Read/Write Access and an error has occurred on the previous access. If Nexus is enabled, this signal is asserted upon unsuccessful (with error) completion of an AHB system bus transfer (Nexus read or write) & is held asserted until the JTAG/OnCE state machine reaches the "Capture_DR" state. Upon exit from system reset or if Nexus is disabled, **nex_err_b** remains deasserted.
nex_evto_b	O	Event Out (**nex_evto_b**) is an output that, when asserted, indicates one of two events has occurred based on the EOC bits in the DC1 Register. **nex_evto_b** is held asserted for one (1) cycle of **nex_mcko**: 1. One (or more) watchpoints has occurred (from Nexus1) & EOC = 2'b00 2. Debug mode was entered (jd_debug_b asserted from Nexus1) & EOC = 2'b01
nex_evti_b	I	Event In (**nex_evti_b**) is an input that, when asserted, will initiate one of two events based on the EIC bits in the DC1 Register (if the Nexus module is enabled at reset): 1. Program Trace & Data Trace synchronization messages (provided Program Trace & Data Trace are enabled & EIC = 2'b00). 2. Debug request to Nexus1 module (provided EIC = 2'b01 and this feature is implemented).
nex_ext_src_id[0:3]	I	**nex_ext_src_id[0:3]** is used to provide the SRC field value used in each message. These pins are tied to a predetermined value at SoC integration time.
nex_sfwcntl_en]	I	**nex_sfwcntl_en** is used to allow software to control the module resources via the DCR register mappings. SoC logic should drive this signal appropriately in a semi-static manner based on the presence of an external hardware debugger and appropriate security precautions.

The Nexus auxiliary port arbitration pins are used when the Nexus 3 module is implemented in a multi-Nexus SoC that shares a single auxiliary output port. The arbitration is controlled by an SoC level Nexus Aurora Router (NAR). Refer to Auxiliary port arbitration for detail on Nexus port arbitration.

Table 67-40. Nexus port arbitration signals

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nex_aux_req[1:0]	O	Nexus Auxiliary Request (**nex_aux_req[1:0]**) output signals indicate to an SoC level Nexus arbiter a request for access to the shared Nexus auxiliary port in a multi-Nexus implementation. The priority encodings are determined by how many messages are currently in the message queues (See Table 67-42.)
nex_aux_busy	O	Nexus Auxiliary Busy (**nex_aux_busy**) is an output signal to an SoC level Nexus arbiter indicating that the Nexus 3 module is currently transmitting its message after being granted the Nexus auxiliary port.

Table continues on the next page...

Table 67-40. Nexus port arbitration signals (continued)

Nexus port arbitration pins	Input/ Output	Description of arbitration pins
nar_aux_grant	I	Nexus Auxiliary Grant **(nar_aux_grant)** is an input from the SoC level NAR that the auxiliary port has been granted to the Nexus 3 module to transmit its message.
ext_multi_nex_sel	I	Multi-Nexus Select (**ext_multi_nex_sel**) is a static signal indicating that the Nexus 3 module is implemented within a multi-Nexus environment. If set, port control and arbitration is controlled by the SoC level arbitration module (NAR).

67.17.2 Pin protocol

The protocol for the processor transmitting messages via the auxiliary pins is accomplished with the MSEO pin function outlined in Table 67-41. Both single and dual pin cases are shown.

nex_mseo_b[1:0] is used to signal the end of variable-length packets, and not fixed length packets. **nex_mseo_b[1:0]** is sampled on the rising edge of the Nexus 3 clock (**nex_mcko**).

Single pin MSEO is not supported on the device.

Table 67-41. MSEO pin(s) protocol

nex_mseo_b function	Single nex_mseo_b data (serial)	Dual nex_mseo_b[1:0] data
Start of message	1-1-0	11-00
End of message	0-1-1-(more 1's)	00 (or 01)-11-(more 1's)
End of variable length packet	0-1-0	00-01
Message transmission	0's	00's
Idle (no message)	1's	11's

Figure 67-54 illustrates the state diagram for single pin MSEO transfers.

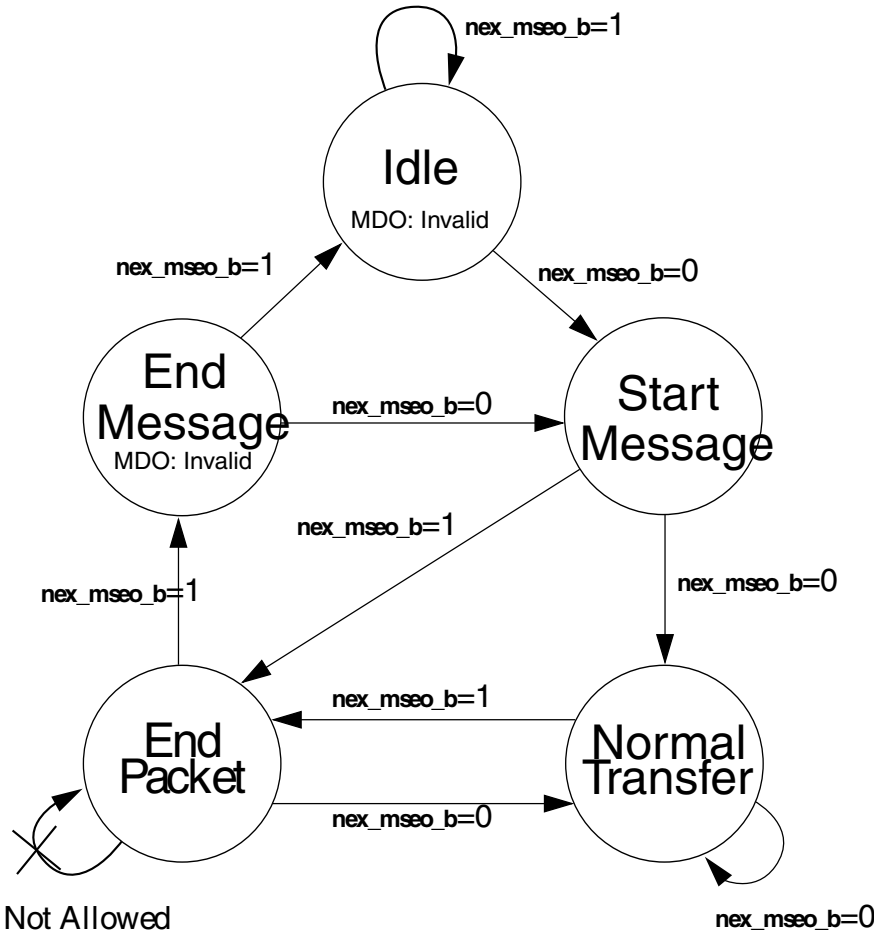


Figure 67-54. Single pin MSEO transfers

Note that the "End Message" state does not contain valid data on the **nex_mdo[n:0]** pins. Also, It is not possible to have two consecutive "End Packet" messages. This implies the minimum packet size for a variable length packet is 2x the number of **nex_mdo[n:0]** pins. This ensures that a false end of message state is not entered by emitting two consecutive '1's on the **nex_mseo_b** pin before the actual end of message.

Figure 67-55 illustrates the state diagram for dual pin MSEO transfers.

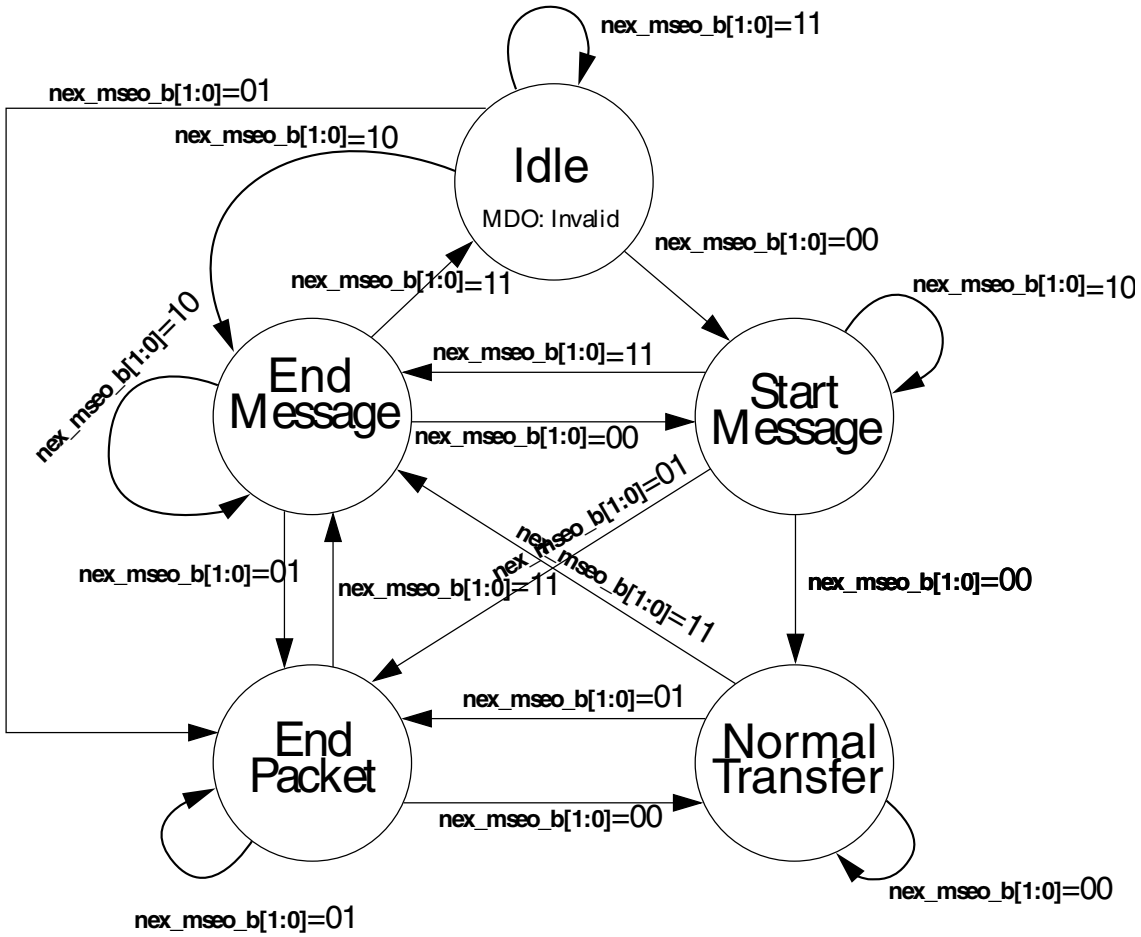


Figure 67-55. Dual pin MSEO transfers

The dual pin MSEO option is more robust that the single pin option. Termination of the current message may immediately be followed by the start of the next message on the consecutive clocks. An extra clock to end the message is not necessary as with the one MSEO pin option. The dual pin option also allows for consecutive "End Packet" states. This can be an advantage when small, variable sized packets are transferred.

Note

The "End Message" state may also indicate the end of a variable-length packet as well as the end of the message when using the dual pin option.

67.18 Rules for output messages

Class 3 compliant embedded processors must provide messages via the auxiliary port in a consistent manner as described below:

- A variable-sized packet within a message must end on a port boundary.
- A variable-sized packet may start within a port boundary only when following a fixed length packet. (If two variable-sized packets end and start on the same clock, it is impossible to know which bit is from the last packet and which bit is from the next packet.)
- Whenever a variable-length packet is sized such that it does not end on a port boundary, it is necessary to extend and zero fill the remaining bits after the highest-order bit so that it can end on a port boundary.

  For example, if the **nex_mdo[n:0]** port is 2 bits wide, and the unique portion of an indirect address TCODE is 5 bits, then the remaining 1 bit of **nex_mdo[n:0]** must be packed with a 0.

67.19 Auxiliary port arbitration

In a multi-Nexus environment, the Nexus 3 module must arbitrate for the shared Nexus port at the SoC level.The request scheme is implemented as a 2-bit request with various levels of priority. The priority levels are defined in Table 67-42 below. The Nexus 3 module will receive a 1-bit grant signal (**nar_aux_grant**) from the SoC level arbiter. When a grant is received, the Nexus 3 module will begin transmitting its message following the protocol outlined in Pin protocol. The Nexus 3 module will maintain control of the port, by asserting the **nex_aux_busy** signal, until the $\overline{\text{MSEO}}$ state machine reaches the "End Message" state.

Table 67-42. MDO Request Encodings

Request level	MDO request encoding (nex_aux_req[1:0])	Condition of queue
No Request	00	No message to send
Low Priority	01	Message queue less than 1/2 full
—	10	Reserved
High Priority	11	Message queue 1/2 full or more

67.20 Examples

The following are examples of Program Trace and Data Trace Messages.

Table 67-43 illustrates an example Indirect Branch Message with 2 MDO / 1MSEO configuration. Table 67-44 illustrates the same example with an 8 MDO / 2 MSEO configuration.

Note that T0 and S0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Ix = Number of instructions (variable)
- Ax = Unique portion of the address (variable)

Note that during clock 13, the **nex_mdo[n:0]** pins are ignored in the single MSEO case.

Table 67-43. Indirect Branch Message Example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I0	MAP	0	Normal Transfer
7	I2	I1	0	Normal Transfer
8	I4	I3	1	End Packet
9	A1	A0	0	Normal Transfer
10	A3	A2	0	Normal Transfer
11	A5	A4	0	Normal Transfer
12	A7	A6	1	End Packet
13	0	0	1	End Message
14	T1	T0	0	Start Message

Table 67-44. Indirect branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	I4	I3	I2	I1	I0	M A	S3	S2	0	1	End Packet

Table continues on the next page...

Table 67-44. Indirect branch message example (8 MDO / 2 MSEO) (continued)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
						P					
3	A7	A6	A5	A4	A3	A2	A1	A0	1	1	End Packet/End Message
4	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 67-45 and Table 67-46 illustrate examples of Direct Branch Messages: one with 2 MDO / 1 MSEO, and one with 8 MDO / 2 MSEO.

Note that T0 and I0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- Ix = Number of Instructions (variable)

Table 67-45. Direct branch message example (2 MDO / 1 MSEO)

Clock	nex_mdo[1:0]		nex_mseo_b	State
0	X	X	1	Idle (or end of last message)
1	T1	T0	0	Start Message
2	T3	T2	0	Normal Transfer
3	T5	T4	0	Normal Transfer
4	S1	S0	0	Normal Transfer
5	S3	S2	0	Normal Transfer
6	I1	I0	1	End Packet
7	0	0	1	End Message

Table 67-46. Direct branch message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	0	0	0	0	I1	I0	S3	S2	1	1	End Packet/End Message
3	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message

Table 67-47 illustrates an example Data Write Message with 8 MDO / 1 MSEO configuration, and Table 67-48 illustrates the same DWM with 8 MDO / 2 MSEO configuration

Note that T0, A0, D0 are the least significant bits where:

- Tx = TCODE number (fixed)
- Sx = Source processor (fixed)
- MAP = (always set to 0)
- Zx = Data size (fixed)
- Ax = Unique portion of the address (variable)
- Dx = Write data (variable 8-, 16- or 32-bit)

Table 67-47. Data write message example (8 MDO / 1 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b	State
0	X	X	X	X	X	X	X	X	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	0	Normal Transfer
4	0	0	0	0	0	0	0	0	1	End Packet
5	0	0	0	0	0	0	0	0	1	End Message

Table 67-48. Data write message example (8 MDO / 2 MSEO)

Clock	nex_mdo[7:0]								nex_mseo_b[1:0]		State
0	X	X	X	X	X	X	X	X	1	1	Idle (or end of last message)
1	S1	S0	T5	T4	T3	T2	T1	T0	0	0	Start Message
2	A0	Z3	Z2	Z1	Z0	0	S3	S2	0	1	End Packet
3	D7	D6	D5	D4	D3	D2	D1	D0	1	1	End Packet/ End Message

67.21 Electrical characteristics

For all electrical characteristics related to core processor and Nexus 3 operation, please refer to the device Data Sheet.

67.22 IEEE 1149.1 (JTAG) RD/WR sequences

This section contains example JTAG/OnCE sequences used to access resources.

67.22.1 JTAG sequence for accessing internal Nexus registers

Table 67-49. Accessing internal Nexus 3 registers via JTAG/OnCE

Step #	TMS Pin	Description
1	1	IDLE -> SELECT-DR_SCAN
2	0	SELECT-DR_SCAN -> CAPTURE-DR (Nexus Command Register value loaded in shifter)
3	0	CAPTURE-DR -> SHIFT-DR
4	0	(7) TCK clocks issued to shift in direction (rd/wr) bit and first 6 bits of Nexus reg. addr.
5	1	SHIFT-DR -> EXIT1-DR (7th bit of Nexus reg. shifted in)
6	1	EXIT1-DR -> UPDATE-DR (Nexus shifter is transferred to Nexus Command Register)
7	1	UPDATE-DR -> SELECT-DR_SCAN
8	0	SELECT-DR_SCAN -> CAPTURE-DR (Register value is transferred to Nexus shifter)
9	0	CAPTURE-DR -> SHIFT-DR
10	0	(31) TCK clocks issued to transfer register value to TDO pin while shifting in TDI value
11	1	SHIFT-DR -> EXIT1-DR (MSB of value is shifted in/out of shifter)
12	1	EXIT1-DR -> UPDATE -DR (if access is write, shifter is transferred to register)
13	0	UPDATE-DR -> RUN-TEST/IDLE (transfer complete - Nexus controller to Reg. Select state)

67.22.2 JTAG sequence for read access of memory-mapped resources

Table 67-50. Accessing memory-mapped resources (reads)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Address Register (RWA)
2	37	Write RWA (initialize starting read address — data input on TDI)
3	13	Nexus Command = write to Read/Write Control/Status Register (RWCS)
4	37	Write RWCS (initialize read access mode and CNT value — data input on TDI)
5	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
6	13	Nexus Command = read Read/Write Access Data Register (RWD)
7	37	Read RWD (data output on TDO)
8	—	If CNT > 0, go back to Step #5

67.22.3 JTAG sequence for write access of memory-mapped resources

Table 67-51. Accessing memory-mapped resources (writes)

Step #	TCLK clocks	Description
1	13	Nexus Command = write to Read/Write Access Control/Status Register (RWCS)
2	37	Write RWCS (initialize write access mode and CNT value - data input on TDI)
3	13	Nexus Command = write to Read/Write Address Register (RWA)
4	37	Write RWA (initialize starting write address - data input on TDI)
5	13	Nexus Command = read Read/Write Access Data Register (RWD)
6	37	Write RWD (data output on TDO)
7	—	Wait for falling edge of **nex_rdy_b** or **nex_err_b** pin
8	—	If CNT > 0, go back to Step #5

Chapter 68
Enhanced Direct Memory Access (eDMA)

68.1 Introduction

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:
  - Source address and destination address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 32 channels

68.1.1 eDMA system block diagram

Figure 68-1 illustrates the components of the eDMA system, including the eDMA module ("engine").

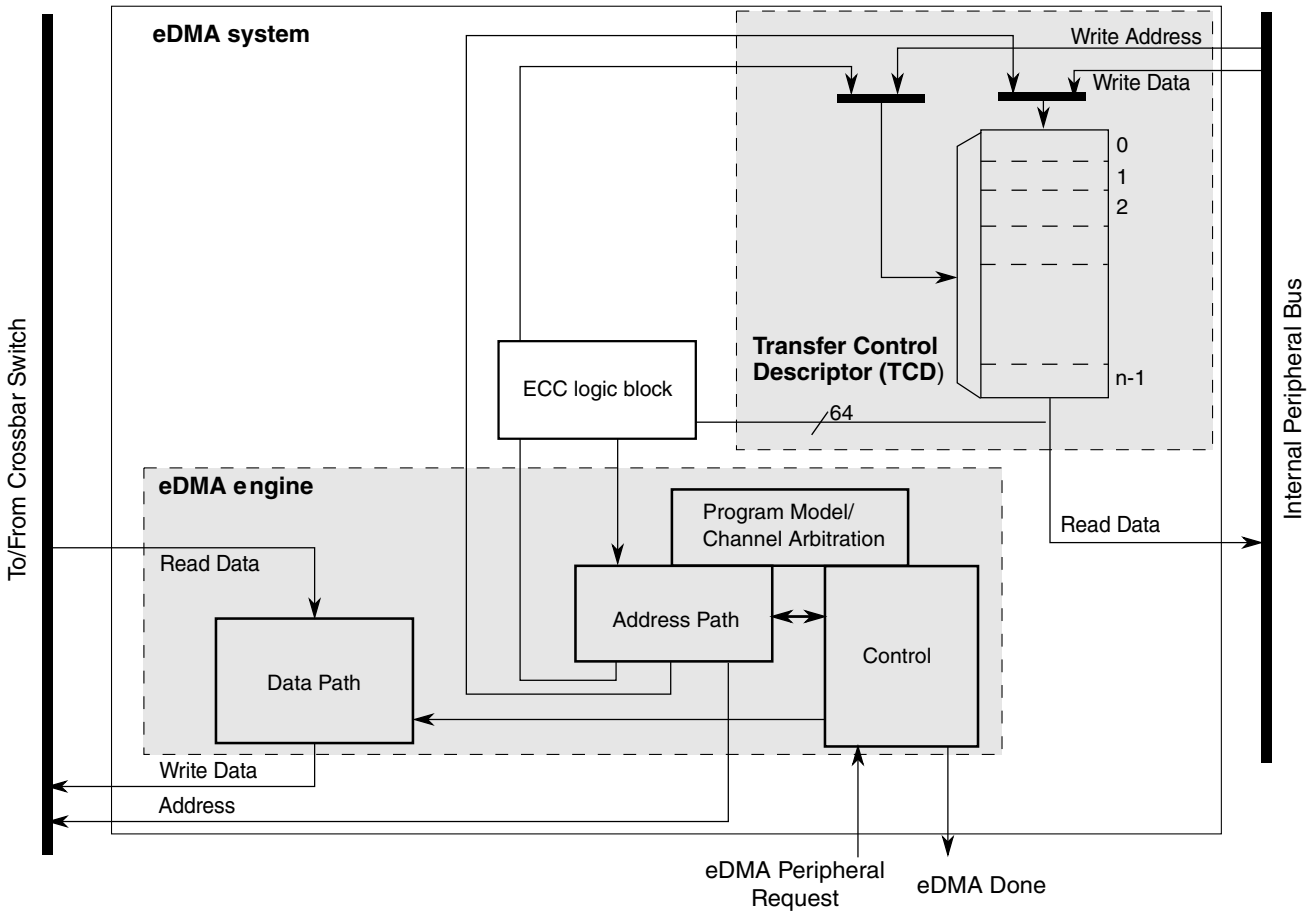


Figure 68-1. eDMA system block diagram

68.1.2 Block parts

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer-control descriptor local memory.

The eDMA engine is further partitioned into four submodules:

Table 68-1. eDMA engine submodules

Submodule	Function
Address path	This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution. When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes

Table continues on the next page...

Table 68-1. eDMA engine submodules (continued)

Submodule	Function
	the new values for the TCD*n*_{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD*n*_CITER field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.
Data path	This block implements the bus master read/write datapath. It includes a data buffer and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output. The address and data path modules directly support the 2-stage pipelined internal bus. The address path module represents the 1st stage of the bus pipeline (address phase), while the data path module implements the 2nd stage of the pipeline (data phase).
Program model/channel arbitration	This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic).
Control	This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/ destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

The transfer-control descriptor local memory is further partitioned into:

Table 68-2. Transfer control descriptor memory

Submodule	Description
Memory controller	This logic implements the required dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
Memory array	TCD storage for each channel's transfer profile.

68.1.3 Features

The eDMA is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the transferred data itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source and destination addresses and transfer size
  - Support for enhanced addressing modes

- 32-channel implementation that performs complex data transfers with minimal intervention from a host processor
  - Internal data buffer, used as temporary storage to support 16- and 32-byte transfers
  - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations
  - 32-byte TCD stored in local memory for each channel
  - An inner data transfer loop defined by a minor byte transfer count
  - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
  - Peripheral-paced hardware requests, one per channel
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via programmable interrupt requests
  - One interrupt per channel, which can be asserted at completion of major iteration count
  - Programmable error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Programmable support for scatter/gather DMA processing
- Support for complex data structures
- Error detection and error correction

In the discussion of this module, *n* is used to reference the channel number.

68.2 Modes of operation

The eDMA operates in the following modes:

Table 68-3. Modes of operation

Mode	Description
Normal	In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA. A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data.
Debug	DMA operation is configurable in Debug mode via the control register: • If CR[EDBG] is cleared, the DMA continues to operate. • If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires.

68.3 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions
- The second region corresponds to the local transfer control descriptor (TCD) memory

68.3.1 TCD memory

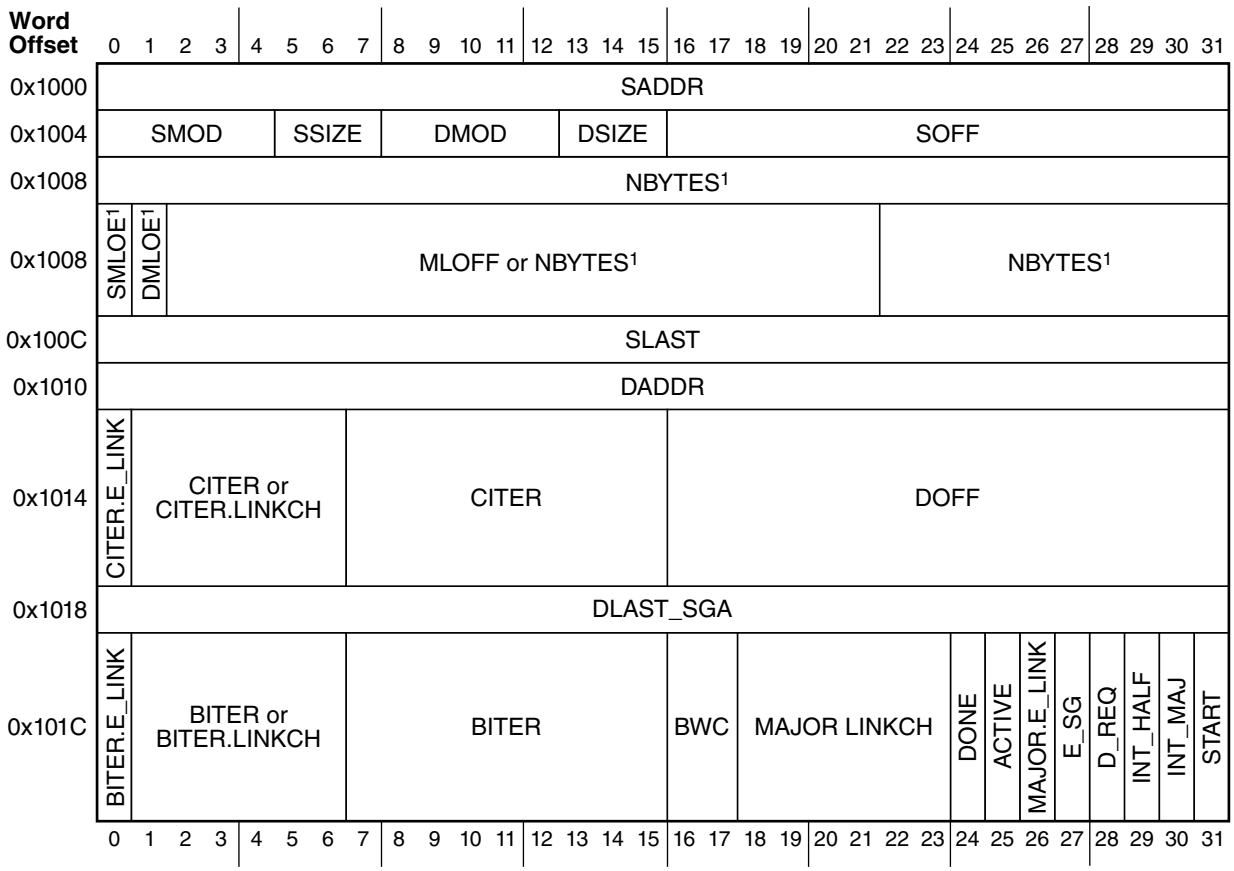
Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1, ... channel 31. Each TCD*n* definition is presented as 11 registers of 16 or 32 bits.

68.3.2 TCD initialization

The initialization of the TCD memory is performed by the eDMA controller after reset. The initialization runs for 256 eDMA clock cycles. All fields in the TCD memory are initialized to 0. All application read or write accesses to the TCD are delayed until the initialization is finished. Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

68.3.3 TCD structure

DMA Basics: TCD Structure
- One DMA engine has a number of channels to react to DMA requests
- Each channel has its own TCD



[1] The fields implemented in Word 2 depend on whether EDMA_CR[EMLM] bit is set to 0 or 1.

68.3.4 Reserved memory and bit fields

- Reading reserved bits in a register returns the value of zero.
- Writes to reserved bits in a register are ignored.
- Reading or writing a reserved memory location generates a bus error.

DMA memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control Register (DMA_CR)	32	R/W	0000_0400h	68.3.5/2889
4	Error Status Register (DMA_ES)	32	R	0000_0000h	68.3.6/2892
C	Enable Request Register (DMA_ERQ)	32	R/W	0000_0000h	68.3.7/2895

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
14	Enable Error Interrupt Register (DMA_EEI)	32	R/W	0000_0000h	68.3.8/2898
18	Set Enable Request Register (DMA_SERQ)	8	W (always reads 0)	00h	68.3.9/2902
19	Clear Enable Request Register (DMA_CERQ)	8	W (always reads 0)	00h	68.3.10/ 2903
1A	Set Enable Error Interrupt Register (DMA_SEEI)	8	W (always reads 0)	00h	68.3.11/ 2904
1B	Clear Enable Error Interrupt Register (DMA_CEEI)	8	W (always reads 0)	00h	68.3.12/ 2905
1C	Clear Interrupt Request Register (DMA_CINT)	8	W (always reads 0)	00h	68.3.13/ 2906
1D	Clear Error Register (DMA_CERR)	8	W (always reads 0)	00h	68.3.14/ 2907
1E	Set START Bit Register (DMA_SSRT)	8	W (always reads 0)	00h	68.3.15/ 2908
1F	Clear DONE Status Bit Register (DMA_CDNE)	8	W (always reads 0)	00h	68.3.16/ 2909
24	Interrupt Request Register (DMA_INT)	32	R/W	0000_0000h	68.3.17/ 2910
2C	Error Register (DMA_ERR)	32	R/W	0000_0000h	68.3.18/ 2913
34	Hardware Request Status Register (DMA_HRS)	32	R	0000_0000h	68.3.19/ 2917
100	Channel n Priority Register (DMA_DCHPRI0)	8	R/W	See section	68.3.20/ 2923
101	Channel n Priority Register (DMA_DCHPRI1)	8	R/W	See section	68.3.20/ 2923
102	Channel n Priority Register (DMA_DCHPRI2)	8	R/W	See section	68.3.20/ 2923
103	Channel n Priority Register (DMA_DCHPRI3)	8	R/W	See section	68.3.20/ 2923
104	Channel n Priority Register (DMA_DCHPRI4)	8	R/W	See section	68.3.20/ 2923
105	Channel n Priority Register (DMA_DCHPRI5)	8	R/W	See section	68.3.20/ 2923
106	Channel n Priority Register (DMA_DCHPRI6)	8	R/W	See section	68.3.20/ 2923

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
107	Channel n Priority Register (DMA_DCHPRI7)	8	R/W	See section	68.3.20/ 2923
108	Channel n Priority Register (DMA_DCHPRI8)	8	R/W	See section	68.3.20/ 2923
109	Channel n Priority Register (DMA_DCHPRI9)	8	R/W	See section	68.3.20/ 2923
10A	Channel n Priority Register (DMA_DCHPRI10)	8	R/W	See section	68.3.20/ 2923
10B	Channel n Priority Register (DMA_DCHPRI11)	8	R/W	See section	68.3.20/ 2923
10C	Channel n Priority Register (DMA_DCHPRI12)	8	R/W	See section	68.3.20/ 2923
10D	Channel n Priority Register (DMA_DCHPRI13)	8	R/W	See section	68.3.20/ 2923
10E	Channel n Priority Register (DMA_DCHPRI14)	8	R/W	See section	68.3.20/ 2923
10F	Channel n Priority Register (DMA_DCHPRI15)	8	R/W	See section	68.3.20/ 2923
110	Channel n Priority Register (DMA_DCHPRI16)	8	R/W	See section	68.3.20/ 2923
111	Channel n Priority Register (DMA_DCHPRI17)	8	R/W	See section	68.3.20/ 2923
112	Channel n Priority Register (DMA_DCHPRI18)	8	R/W	See section	68.3.20/ 2923
113	Channel n Priority Register (DMA_DCHPRI19)	8	R/W	See section	68.3.20/ 2923
114	Channel n Priority Register (DMA_DCHPRI20)	8	R/W	See section	68.3.20/ 2923
115	Channel n Priority Register (DMA_DCHPRI21)	8	R/W	See section	68.3.20/ 2923
116	Channel n Priority Register (DMA_DCHPRI22)	8	R/W	See section	68.3.20/ 2923
117	Channel n Priority Register (DMA_DCHPRI23)	8	R/W	See section	68.3.20/ 2923
118	Channel n Priority Register (DMA_DCHPRI24)	8	R/W	See section	68.3.20/ 2923
119	Channel n Priority Register (DMA_DCHPRI25)	8	R/W	See section	68.3.20/ 2923
11A	Channel n Priority Register (DMA_DCHPRI26)	8	R/W	See section	68.3.20/ 2923
11B	Channel n Priority Register (DMA_DCHPRI27)	8	R/W	See section	68.3.20/ 2923
11C	Channel n Priority Register (DMA_DCHPRI28)	8	R/W	See section	68.3.20/ 2923

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11D	Channel n Priority Register (DMA_DCHPRI29)	8	R/W	See section	68.3.20/ 2923
11E	Channel n Priority Register (DMA_DCHPRI30)	8	R/W	See section	68.3.20/ 2923
11F	Channel n Priority Register (DMA_DCHPRI31)	8	R/W	See section	68.3.20/ 2923
140	Channel n Master ID Register (DMA_DCHMID0)	8	R/W	00h	68.3.21/ 2924
141	Channel n Master ID Register (DMA_DCHMID1)	8	R/W	00h	68.3.21/ 2924
142	Channel n Master ID Register (DMA_DCHMID2)	8	R/W	00h	68.3.21/ 2924
143	Channel n Master ID Register (DMA_DCHMID3)	8	R/W	00h	68.3.21/ 2924
144	Channel n Master ID Register (DMA_DCHMID4)	8	R/W	00h	68.3.21/ 2924
145	Channel n Master ID Register (DMA_DCHMID5)	8	R/W	00h	68.3.21/ 2924
146	Channel n Master ID Register (DMA_DCHMID6)	8	R/W	00h	68.3.21/ 2924
147	Channel n Master ID Register (DMA_DCHMID7)	8	R/W	00h	68.3.21/ 2924
148	Channel n Master ID Register (DMA_DCHMID8)	8	R/W	00h	68.3.21/ 2924
149	Channel n Master ID Register (DMA_DCHMID9)	8	R/W	00h	68.3.21/ 2924
14A	Channel n Master ID Register (DMA_DCHMID10)	8	R/W	00h	68.3.21/ 2924
14B	Channel n Master ID Register (DMA_DCHMID11)	8	R/W	00h	68.3.21/ 2924
14C	Channel n Master ID Register (DMA_DCHMID12)	8	R/W	00h	68.3.21/ 2924
14D	Channel n Master ID Register (DMA_DCHMID13)	8	R/W	00h	68.3.21/ 2924
14E	Channel n Master ID Register (DMA_DCHMID14)	8	R/W	00h	68.3.21/ 2924
14F	Channel n Master ID Register (DMA_DCHMID15)	8	R/W	00h	68.3.21/ 2924
150	Channel n Master ID Register (DMA_DCHMID16)	8	R/W	00h	68.3.21/ 2924
151	Channel n Master ID Register (DMA_DCHMID17)	8	R/W	00h	68.3.21/ 2924
152	Channel n Master ID Register (DMA_DCHMID18)	8	R/W	00h	68.3.21/ 2924

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
153	Channel n Master ID Register (DMA_DCHMID19)	8	R/W	00h	68.3.21/ 2924
154	Channel n Master ID Register (DMA_DCHMID20)	8	R/W	00h	68.3.21/ 2924
155	Channel n Master ID Register (DMA_DCHMID21)	8	R/W	00h	68.3.21/ 2924
156	Channel n Master ID Register (DMA_DCHMID22)	8	R/W	00h	68.3.21/ 2924
157	Channel n Master ID Register (DMA_DCHMID23)	8	R/W	00h	68.3.21/ 2924
158	Channel n Master ID Register (DMA_DCHMID24)	8	R/W	00h	68.3.21/ 2924
159	Channel n Master ID Register (DMA_DCHMID25)	8	R/W	00h	68.3.21/ 2924
15A	Channel n Master ID Register (DMA_DCHMID26)	8	R/W	00h	68.3.21/ 2924
15B	Channel n Master ID Register (DMA_DCHMID27)	8	R/W	00h	68.3.21/ 2924
15C	Channel n Master ID Register (DMA_DCHMID28)	8	R/W	00h	68.3.21/ 2924
15D	Channel n Master ID Register (DMA_DCHMID29)	8	R/W	00h	68.3.21/ 2924
15E	Channel n Master ID Register (DMA_DCHMID30)	8	R/W	00h	68.3.21/ 2924
15F	Channel n Master ID Register (DMA_DCHMID31)	8	R/W	00h	68.3.21/ 2924
1000	TCD Source Address (DMA_TCD0_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1004	TCD Transfer Attributes (DMA_TCD0_ATTR)	16	R/W	0000h	68.3.23/ 2925
1006	TCD Signed Source Address Offset (DMA_TCD0_SOFF)	16	R/W	0000h	68.3.24/ 2926
1008	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD0_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1008	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD0_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1008	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD0_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
100C	TCD Last Source Address Adjustment (DMA_TCD0_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1010	TCD Destination Address (DMA_TCD0_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1014	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1014	DMA_TCD0_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1016	TCD Signed Destination Address Offset (DMA_TCD0_DOFF)	16	R/W	0000h	68.3.32/ 2933
1018	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD0_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
101C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD0_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
101C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD0_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
101E	TCD Control and Status (DMA_TCD0_CSR)	16	R/W	0000h	68.3.36/ 2936
1020	TCD Source Address (DMA_TCD1_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1024	TCD Transfer Attributes (DMA_TCD1_ATTR)	16	R/W	0000h	68.3.23/ 2925
1026	TCD Signed Source Address Offset (DMA_TCD1_SOFF)	16	R/W	0000h	68.3.24/ 2926
1028	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD1_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1028	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD1_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1028	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD1_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
102C	TCD Last Source Address Adjustment (DMA_TCD1_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1030	TCD Destination Address (DMA_TCD1_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1034	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1034	DMA_TCD1_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1036	TCD Signed Destination Address Offset (DMA_TCD1_DOFF)	16	R/W	0000h	68.3.32/ 2933
1038	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD1_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
103C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD1_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
103C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD1_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
103E	TCD Control and Status (DMA_TCD1_CSR)	16	R/W	0000h	68.3.36/ 2936

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1040	TCD Source Address (DMA_TCD2_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1044	TCD Transfer Attributes (DMA_TCD2_ATTR)	16	R/W	0000h	68.3.23/ 2925
1046	TCD Signed Source Address Offset (DMA_TCD2_SOFF)	16	R/W	0000h	68.3.24/ 2926
1048	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD2_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1048	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD2_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1048	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD2_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
104C	TCD Last Source Address Adjustment (DMA_TCD2_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1050	TCD Destination Address (DMA_TCD2_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1054	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1054	DMA_TCD2_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1056	TCD Signed Destination Address Offset (DMA_TCD2_DOFF)	16	R/W	0000h	68.3.32/ 2933
1058	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD2_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
105C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD2_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
105C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD2_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
105E	TCD Control and Status (DMA_TCD2_CSR)	16	R/W	0000h	68.3.36/ 2936
1060	TCD Source Address (DMA_TCD3_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1064	TCD Transfer Attributes (DMA_TCD3_ATTR)	16	R/W	0000h	68.3.23/ 2925
1066	TCD Signed Source Address Offset (DMA_TCD3_SOFF)	16	R/W	0000h	68.3.24/ 2926
1068	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD3_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1068	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD3_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1068	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD3_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
106C	TCD Last Source Address Adjustment (DMA_TCD3_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1070	TCD Destination Address (DMA_TCD3_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1074	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1074	DMA_TCD3_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1076	TCD Signed Destination Address Offset (DMA_TCD3_DOFF)	16	R/W	0000h	68.3.32/ 2933
1078	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD3_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
107C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD3_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
107C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD3_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
107E	TCD Control and Status (DMA_TCD3_CSR)	16	R/W	0000h	68.3.36/ 2936
1080	TCD Source Address (DMA_TCD4_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1084	TCD Transfer Attributes (DMA_TCD4_ATTR)	16	R/W	0000h	68.3.23/ 2925
1086	TCD Signed Source Address Offset (DMA_TCD4_SOFF)	16	R/W	0000h	68.3.24/ 2926
1088	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD4_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1088	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD4_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1088	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD4_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
108C	TCD Last Source Address Adjustment (DMA_TCD4_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1090	TCD Destination Address (DMA_TCD4_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1094	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1094	DMA_TCD4_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1096	TCD Signed Destination Address Offset (DMA_TCD4_DOFF)	16	R/W	0000h	68.3.32/ 2933
1098	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD4_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
109C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD4_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
109C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD4_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
109E	TCD Control and Status (DMA_TCD4_CSR)	16	R/W	0000h	68.3.36/ 2936
10A0	TCD Source Address (DMA_TCD5_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
10A4	TCD Transfer Attributes (DMA_TCD5_ATTR)	16	R/W	0000h	68.3.23/ 2925
10A6	TCD Signed Source Address Offset (DMA_TCD5_SOFF)	16	R/W	0000h	68.3.24/ 2926
10A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD5_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
10A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD5_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
10A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD5_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
10AC	TCD Last Source Address Adjustment (DMA_TCD5_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
10B0	TCD Destination Address (DMA_TCD5_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
10B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
10B4	DMA_TCD5_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
10B6	TCD Signed Destination Address Offset (DMA_TCD5_DOFF)	16	R/W	0000h	68.3.32/ 2933
10B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD5_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
10BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD5_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
10BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD5_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
10BE	TCD Control and Status (DMA_TCD5_CSR)	16	R/W	0000h	68.3.36/ 2936
10C0	TCD Source Address (DMA_TCD6_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
10C4	TCD Transfer Attributes (DMA_TCD6_ATTR)	16	R/W	0000h	68.3.23/ 2925
10C6	TCD Signed Source Address Offset (DMA_TCD6_SOFF)	16	R/W	0000h	68.3.24/ 2926

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD6_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
10C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD6_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
10C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD6_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
10CC	TCD Last Source Address Adjustment (DMA_TCD6_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
10D0	TCD Destination Address (DMA_TCD6_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
10D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
10D4	DMA_TCD6_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
10D6	TCD Signed Destination Address Offset (DMA_TCD6_DOFF)	16	R/W	0000h	68.3.32/ 2933
10D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD6_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
10DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD6_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
10DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD6_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
10DE	TCD Control and Status (DMA_TCD6_CSR)	16	R/W	0000h	68.3.36/ 2936
10E0	TCD Source Address (DMA_TCD7_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
10E4	TCD Transfer Attributes (DMA_TCD7_ATTR)	16	R/W	0000h	68.3.23/ 2925
10E6	TCD Signed Source Address Offset (DMA_TCD7_SOFF)	16	R/W	0000h	68.3.24/ 2926
10E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD7_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
10E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD7_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
10E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD7_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
10EC	TCD Last Source Address Adjustment (DMA_TCD7_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
10F0	TCD Destination Address (DMA_TCD7_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
10F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10F4	DMA_TCD7_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
10F6	TCD Signed Destination Address Offset (DMA_TCD7_DOFF)	16	R/W	0000h	68.3.32/ 2933
10F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD7_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
10FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD7_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
10FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD7_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
10FE	TCD Control and Status (DMA_TCD7_CSR)	16	R/W	0000h	68.3.36/ 2936
1100	TCD Source Address (DMA_TCD8_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1104	TCD Transfer Attributes (DMA_TCD8_ATTR)	16	R/W	0000h	68.3.23/ 2925
1106	TCD Signed Source Address Offset (DMA_TCD8_SOFF)	16	R/W	0000h	68.3.24/ 2926
1108	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD8_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1108	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD8_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1108	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD8_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
110C	TCD Last Source Address Adjustment (DMA_TCD8_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1110	TCD Destination Address (DMA_TCD8_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1114	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD8_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1114	DMA_TCD8_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1116	TCD Signed Destination Address Offset (DMA_TCD8_DOFF)	16	R/W	0000h	68.3.32/ 2933
1118	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD8_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
111C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD8_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
111C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD8_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
111E	TCD Control and Status (DMA_TCD8_CSR)	16	R/W	0000h	68.3.36/ 2936

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1120	TCD Source Address (DMA_TCD9_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1124	TCD Transfer Attributes (DMA_TCD9_ATTR)	16	R/W	0000h	68.3.23/ 2925
1126	TCD Signed Source Address Offset (DMA_TCD9_SOFF)	16	R/W	0000h	68.3.24/ 2926
1128	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD9_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1128	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD9_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1128	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD9_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
112C	TCD Last Source Address Adjustment (DMA_TCD9_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1130	TCD Destination Address (DMA_TCD9_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1134	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD9_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1134	DMA_TCD9_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1136	TCD Signed Destination Address Offset (DMA_TCD9_DOFF)	16	R/W	0000h	68.3.32/ 2933
1138	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD9_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
113C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD9_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
113C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD9_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
113E	TCD Control and Status (DMA_TCD9_CSR)	16	R/W	0000h	68.3.36/ 2936
1140	TCD Source Address (DMA_TCD10_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1144	TCD Transfer Attributes (DMA_TCD10_ATTR)	16	R/W	0000h	68.3.23/ 2925
1146	TCD Signed Source Address Offset (DMA_TCD10_SOFF)	16	R/W	0000h	68.3.24/ 2926
1148	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD10_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1148	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD10_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1148	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD10_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
114C	TCD Last Source Address Adjustment (DMA_TCD10_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1150	TCD Destination Address (DMA_TCD10_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1154	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD10_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1154	DMA_TCD10_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1156	TCD Signed Destination Address Offset (DMA_TCD10_DOFF)	16	R/W	0000h	68.3.32/ 2933
1158	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD10_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
115C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD10_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
115C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD10_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
115E	TCD Control and Status (DMA_TCD10_CSR)	16	R/W	0000h	68.3.36/ 2936
1160	TCD Source Address (DMA_TCD11_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1164	TCD Transfer Attributes (DMA_TCD11_ATTR)	16	R/W	0000h	68.3.23/ 2925
1166	TCD Signed Source Address Offset (DMA_TCD11_SOFF)	16	R/W	0000h	68.3.24/ 2926
1168	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD11_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1168	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD11_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1168	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD11_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
116C	TCD Last Source Address Adjustment (DMA_TCD11_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1170	TCD Destination Address (DMA_TCD11_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1174	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD11_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1174	DMA_TCD11_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1176	TCD Signed Destination Address Offset (DMA_TCD11_DOFF)	16	R/W	0000h	68.3.32/ 2933
1178	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD11_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
117C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD11_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
117C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD11_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
117E	TCD Control and Status (DMA_TCD11_CSR)	16	R/W	0000h	68.3.36/ 2936
1180	TCD Source Address (DMA_TCD12_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1184	TCD Transfer Attributes (DMA_TCD12_ATTR)	16	R/W	0000h	68.3.23/ 2925
1186	TCD Signed Source Address Offset (DMA_TCD12_SOFF)	16	R/W	0000h	68.3.24/ 2926
1188	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD12_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1188	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD12_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1188	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD12_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
118C	TCD Last Source Address Adjustment (DMA_TCD12_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1190	TCD Destination Address (DMA_TCD12_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1194	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD12_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1194	DMA_TCD12_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1196	TCD Signed Destination Address Offset (DMA_TCD12_DOFF)	16	R/W	0000h	68.3.32/ 2933
1198	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD12_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
119C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD12_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
119C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD12_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
119E	TCD Control and Status (DMA_TCD12_CSR)	16	R/W	0000h	68.3.36/ 2936
11A0	TCD Source Address (DMA_TCD13_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
11A4	TCD Transfer Attributes (DMA_TCD13_ATTR)	16	R/W	0000h	68.3.23/ 2925

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11A6	TCD Signed Source Address Offset (DMA_TCD13_SOFF)	16	R/W	0000h	68.3.24/ 2926
11A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD13_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
11A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD13_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
11A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD13_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
11AC	TCD Last Source Address Adjustment (DMA_TCD13_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
11B0	TCD Destination Address (DMA_TCD13_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
11B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD13_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
11B4	DMA_TCD13_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
11B6	TCD Signed Destination Address Offset (DMA_TCD13_DOFF)	16	R/W	0000h	68.3.32/ 2933
11B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD13_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
11BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD13_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
11BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD13_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
11BE	TCD Control and Status (DMA_TCD13_CSR)	16	R/W	0000h	68.3.36/ 2936
11C0	TCD Source Address (DMA_TCD14_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
11C4	TCD Transfer Attributes (DMA_TCD14_ATTR)	16	R/W	0000h	68.3.23/ 2925
11C6	TCD Signed Source Address Offset (DMA_TCD14_SOFF)	16	R/W	0000h	68.3.24/ 2926
11C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD14_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
11C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD14_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
11C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD14_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
11CC	TCD Last Source Address Adjustment (DMA_TCD14_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
11D0	TCD Destination Address (DMA_TCD14_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD14_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
11D4	DMA_TCD14_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
11D6	TCD Signed Destination Address Offset (DMA_TCD14_DOFF)	16	R/W	0000h	68.3.32/ 2933
11D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD14_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
11DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD14_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
11DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD14_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
11DE	TCD Control and Status (DMA_TCD14_CSR)	16	R/W	0000h	68.3.36/ 2936
11E0	TCD Source Address (DMA_TCD15_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
11E4	TCD Transfer Attributes (DMA_TCD15_ATTR)	16	R/W	0000h	68.3.23/ 2925
11E6	TCD Signed Source Address Offset (DMA_TCD15_SOFF)	16	R/W	0000h	68.3.24/ 2926
11E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD15_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
11E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD15_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
11E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD15_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
11EC	TCD Last Source Address Adjustment (DMA_TCD15_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
11F0	TCD Destination Address (DMA_TCD15_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
11F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD15_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
11F4	DMA_TCD15_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
11F6	TCD Signed Destination Address Offset (DMA_TCD15_DOFF)	16	R/W	0000h	68.3.32/ 2933
11F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD15_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
11FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD15_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD15_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
11FE	TCD Control and Status (DMA_TCD15_CSR)	16	R/W	0000h	68.3.36/ 2936
1200	TCD Source Address (DMA_TCD16_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1204	TCD Transfer Attributes (DMA_TCD16_ATTR)	16	R/W	0000h	68.3.23/ 2925
1206	TCD Signed Source Address Offset (DMA_TCD16_SOFF)	16	R/W	0000h	68.3.24/ 2926
1208	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD16_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1208	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD16_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1208	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD16_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
120C	TCD Last Source Address Adjustment (DMA_TCD16_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1210	TCD Destination Address (DMA_TCD16_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1214	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD16_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1214	DMA_TCD16_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1216	TCD Signed Destination Address Offset (DMA_TCD16_DOFF)	16	R/W	0000h	68.3.32/ 2933
1218	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD16_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
121C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD16_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
121C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD16_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
121E	TCD Control and Status (DMA_TCD16_CSR)	16	R/W	0000h	68.3.36/ 2936
1220	TCD Source Address (DMA_TCD17_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1224	TCD Transfer Attributes (DMA_TCD17_ATTR)	16	R/W	0000h	68.3.23/ 2925
1226	TCD Signed Source Address Offset (DMA_TCD17_SOFF)	16	R/W	0000h	68.3.24/ 2926
1228	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD17_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1228	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD17_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1228	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD17_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
122C	TCD Last Source Address Adjustment (DMA_TCD17_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1230	TCD Destination Address (DMA_TCD17_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1234	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD17_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1234	DMA_TCD17_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1236	TCD Signed Destination Address Offset (DMA_TCD17_DOFF)	16	R/W	0000h	68.3.32/ 2933
1238	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD17_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
123C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD17_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
123C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD17_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
123E	TCD Control and Status (DMA_TCD17_CSR)	16	R/W	0000h	68.3.36/ 2936
1240	TCD Source Address (DMA_TCD18_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1244	TCD Transfer Attributes (DMA_TCD18_ATTR)	16	R/W	0000h	68.3.23/ 2925
1246	TCD Signed Source Address Offset (DMA_TCD18_SOFF)	16	R/W	0000h	68.3.24/ 2926
1248	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD18_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1248	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD18_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1248	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD18_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
124C	TCD Last Source Address Adjustment (DMA_TCD18_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1250	TCD Destination Address (DMA_TCD18_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1254	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD18_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1254	DMA_TCD18_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1256	TCD Signed Destination Address Offset (DMA_TCD18_DOFF)	16	R/W	0000h	68.3.32/ 2933
1258	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD18_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
125C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD18_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
125C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD18_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
125E	TCD Control and Status (DMA_TCD18_CSR)	16	R/W	0000h	68.3.36/ 2936
1260	TCD Source Address (DMA_TCD19_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1264	TCD Transfer Attributes (DMA_TCD19_ATTR)	16	R/W	0000h	68.3.23/ 2925
1266	TCD Signed Source Address Offset (DMA_TCD19_SOFF)	16	R/W	0000h	68.3.24/ 2926
1268	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD19_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1268	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD19_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1268	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD19_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
126C	TCD Last Source Address Adjustment (DMA_TCD19_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1270	TCD Destination Address (DMA_TCD19_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1274	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD19_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1274	DMA_TCD19_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1276	TCD Signed Destination Address Offset (DMA_TCD19_DOFF)	16	R/W	0000h	68.3.32/ 2933
1278	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD19_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
127C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD19_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
127C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD19_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
127E	TCD Control and Status (DMA_TCD19_CSR)	16	R/W	0000h	68.3.36/ 2936

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1280	TCD Source Address (DMA_TCD20_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1284	TCD Transfer Attributes (DMA_TCD20_ATTR)	16	R/W	0000h	68.3.23/ 2925
1286	TCD Signed Source Address Offset (DMA_TCD20_SOFF)	16	R/W	0000h	68.3.24/ 2926
1288	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD20_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1288	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD20_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1288	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD20_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
128C	TCD Last Source Address Adjustment (DMA_TCD20_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1290	TCD Destination Address (DMA_TCD20_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1294	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD20_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1294	DMA_TCD20_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1296	TCD Signed Destination Address Offset (DMA_TCD20_DOFF)	16	R/W	0000h	68.3.32/ 2933
1298	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD20_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
129C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD20_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
129C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD20_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
129E	TCD Control and Status (DMA_TCD20_CSR)	16	R/W	0000h	68.3.36/ 2936
12A0	TCD Source Address (DMA_TCD21_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
12A4	TCD Transfer Attributes (DMA_TCD21_ATTR)	16	R/W	0000h	68.3.23/ 2925
12A6	TCD Signed Source Address Offset (DMA_TCD21_SOFF)	16	R/W	0000h	68.3.24/ 2926
12A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD21_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
12A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD21_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
12A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD21_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12AC	TCD Last Source Address Adjustment (DMA_TCD21_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
12B0	TCD Destination Address (DMA_TCD21_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
12B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD21_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
12B4	DMA_TCD21_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
12B6	TCD Signed Destination Address Offset (DMA_TCD21_DOFF)	16	R/W	0000h	68.3.32/ 2933
12B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD21_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
12BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD21_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
12BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD21_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
12BE	TCD Control and Status (DMA_TCD21_CSR)	16	R/W	0000h	68.3.36/ 2936
12C0	TCD Source Address (DMA_TCD22_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
12C4	TCD Transfer Attributes (DMA_TCD22_ATTR)	16	R/W	0000h	68.3.23/ 2925
12C6	TCD Signed Source Address Offset (DMA_TCD22_SOFF)	16	R/W	0000h	68.3.24/ 2926
12C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD22_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
12C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD22_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
12C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD22_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
12CC	TCD Last Source Address Adjustment (DMA_TCD22_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
12D0	TCD Destination Address (DMA_TCD22_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
12D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD22_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
12D4	DMA_TCD22_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
12D6	TCD Signed Destination Address Offset (DMA_TCD22_DOFF)	16	R/W	0000h	68.3.32/ 2933
12D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD22_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD22_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
12DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD22_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
12DE	TCD Control and Status (DMA_TCD22_CSR)	16	R/W	0000h	68.3.36/ 2936
12E0	TCD Source Address (DMA_TCD23_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
12E4	TCD Transfer Attributes (DMA_TCD23_ATTR)	16	R/W	0000h	68.3.23/ 2925
12E6	TCD Signed Source Address Offset (DMA_TCD23_SOFF)	16	R/W	0000h	68.3.24/ 2926
12E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD23_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
12E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD23_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
12E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD23_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
12EC	TCD Last Source Address Adjustment (DMA_TCD23_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
12F0	TCD Destination Address (DMA_TCD23_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
12F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD23_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
12F4	DMA_TCD23_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
12F6	TCD Signed Destination Address Offset (DMA_TCD23_DOFF)	16	R/W	0000h	68.3.32/ 2933
12F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD23_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
12FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD23_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
12FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD23_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
12FE	TCD Control and Status (DMA_TCD23_CSR)	16	R/W	0000h	68.3.36/ 2936
1300	TCD Source Address (DMA_TCD24_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1304	TCD Transfer Attributes (DMA_TCD24_ATTR)	16	R/W	0000h	68.3.23/ 2925

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1306	TCD Signed Source Address Offset (DMA_TCD24_SOFF)	16	R/W	0000h	68.3.24/ 2926
1308	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD24_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1308	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD24_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1308	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD24_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
130C	TCD Last Source Address Adjustment (DMA_TCD24_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1310	TCD Destination Address (DMA_TCD24_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1314	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD24_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1314	DMA_TCD24_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1316	TCD Signed Destination Address Offset (DMA_TCD24_DOFF)	16	R/W	0000h	68.3.32/ 2933
1318	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD24_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
131C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD24_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
131C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD24_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
131E	TCD Control and Status (DMA_TCD24_CSR)	16	R/W	0000h	68.3.36/ 2936
1320	TCD Source Address (DMA_TCD25_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1324	TCD Transfer Attributes (DMA_TCD25_ATTR)	16	R/W	0000h	68.3.23/ 2925
1326	TCD Signed Source Address Offset (DMA_TCD25_SOFF)	16	R/W	0000h	68.3.24/ 2926
1328	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD25_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1328	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD25_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1328	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD25_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
132C	TCD Last Source Address Adjustment (DMA_TCD25_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1330	TCD Destination Address (DMA_TCD25_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1334	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD25_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1334	DMA_TCD25_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1336	TCD Signed Destination Address Offset (DMA_TCD25_DOFF)	16	R/W	0000h	68.3.32/ 2933
1338	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD25_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
133C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD25_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
133C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD25_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
133E	TCD Control and Status (DMA_TCD25_CSR)	16	R/W	0000h	68.3.36/ 2936
1340	TCD Source Address (DMA_TCD26_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1344	TCD Transfer Attributes (DMA_TCD26_ATTR)	16	R/W	0000h	68.3.23/ 2925
1346	TCD Signed Source Address Offset (DMA_TCD26_SOFF)	16	R/W	0000h	68.3.24/ 2926
1348	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD26_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1348	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD26_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1348	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD26_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
134C	TCD Last Source Address Adjustment (DMA_TCD26_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1350	TCD Destination Address (DMA_TCD26_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1354	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD26_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1354	DMA_TCD26_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1356	TCD Signed Destination Address Offset (DMA_TCD26_DOFF)	16	R/W	0000h	68.3.32/ 2933
1358	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD26_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
135C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD26_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
135C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD26_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
135E	TCD Control and Status (DMA_TCD26_CSR)	16	R/W	0000h	68.3.36/ 2936
1360	TCD Source Address (DMA_TCD27_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1364	TCD Transfer Attributes (DMA_TCD27_ATTR)	16	R/W	0000h	68.3.23/ 2925
1366	TCD Signed Source Address Offset (DMA_TCD27_SOFF)	16	R/W	0000h	68.3.24/ 2926
1368	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD27_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
1368	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD27_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1368	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD27_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
136C	TCD Last Source Address Adjustment (DMA_TCD27_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1370	TCD Destination Address (DMA_TCD27_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1374	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD27_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1374	DMA_TCD27_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1376	TCD Signed Destination Address Offset (DMA_TCD27_DOFF)	16	R/W	0000h	68.3.32/ 2933
1378	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD27_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
137C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD27_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
137C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD27_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
137E	TCD Control and Status (DMA_TCD27_CSR)	16	R/W	0000h	68.3.36/ 2936
1380	TCD Source Address (DMA_TCD28_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
1384	TCD Transfer Attributes (DMA_TCD28_ATTR)	16	R/W	0000h	68.3.23/ 2925
1386	TCD Signed Source Address Offset (DMA_TCD28_SOFF)	16	R/W	0000h	68.3.24/ 2926
1388	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD28_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1388	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD28_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
1388	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD28_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
138C	TCD Last Source Address Adjustment (DMA_TCD28_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
1390	TCD Destination Address (DMA_TCD28_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
1394	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD28_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
1394	DMA_TCD28_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
1396	TCD Signed Destination Address Offset (DMA_TCD28_DOFF)	16	R/W	0000h	68.3.32/ 2933
1398	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD28_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
139C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD28_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
139C	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD28_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
139E	TCD Control and Status (DMA_TCD28_CSR)	16	R/W	0000h	68.3.36/ 2936
13A0	TCD Source Address (DMA_TCD29_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
13A4	TCD Transfer Attributes (DMA_TCD29_ATTR)	16	R/W	0000h	68.3.23/ 2925
13A6	TCD Signed Source Address Offset (DMA_TCD29_SOFF)	16	R/W	0000h	68.3.24/ 2926
13A8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD29_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
13A8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD29_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
13A8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD29_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
13AC	TCD Last Source Address Adjustment (DMA_TCD29_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
13B0	TCD Destination Address (DMA_TCD29_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
13B4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD29_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
13B4	DMA_TCD29_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13B6	TCD Signed Destination Address Offset (DMA_TCD29_DOFF)	16	R/W	0000h	68.3.32/ 2933
13B8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD29_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
13BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD29_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
13BC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD29_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
13BE	TCD Control and Status (DMA_TCD29_CSR)	16	R/W	0000h	68.3.36/ 2936
13C0	TCD Source Address (DMA_TCD30_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
13C4	TCD Transfer Attributes (DMA_TCD30_ATTR)	16	R/W	0000h	68.3.23/ 2925
13C6	TCD Signed Source Address Offset (DMA_TCD30_SOFF)	16	R/W	0000h	68.3.24/ 2926
13C8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD30_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
13C8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD30_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
13C8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD30_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
13CC	TCD Last Source Address Adjustment (DMA_TCD30_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
13D0	TCD Destination Address (DMA_TCD30_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
13D4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD30_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
13D4	DMA_TCD30_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
13D6	TCD Signed Destination Address Offset (DMA_TCD30_DOFF)	16	R/W	0000h	68.3.32/ 2933
13D8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD30_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
13DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD30_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
13DC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD30_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
13DE	TCD Control and Status (DMA_TCD30_CSR)	16	R/W	0000h	68.3.36/ 2936

Table continues on the next page...

DMA memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13E0	TCD Source Address (DMA_TCD31_SADDR)	32	R/W	0000_0000h	68.3.22/ 2925
13E4	TCD Transfer Attributes (DMA_TCD31_ATTR)	16	R/W	0000h	68.3.23/ 2925
13E6	TCD Signed Source Address Offset (DMA_TCD31_SOFF)	16	R/W	0000h	68.3.24/ 2926
13E8	TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD31_NBYTES_MLNO)	32	R/W	0000_0000h	68.3.25/ 2927
13E8	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD31_NBYTES_MLOFFNO)	32	R/W	0000_0000h	68.3.26/ 2927
13E8	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD31_NBYTES_MLOFFYES)	32	R/W	0000_0000h	68.3.27/ 2929
13EC	TCD Last Source Address Adjustment (DMA_TCD31_SLAST)	32	R/W	0000_0000h	68.3.28/ 2930
13F0	TCD Destination Address (DMA_TCD31_DADDR)	32	R/W	0000_0000h	68.3.29/ 2930
13F4	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD31_CITER_ELINKYES)	16	R/W	0000h	68.3.30/ 2931
13F4	DMA_TCD31_CITER_ELINKNO	16	R/W	0000h	68.3.31/ 2932
13F6	TCD Signed Destination Address Offset (DMA_TCD31_DOFF)	16	R/W	0000h	68.3.32/ 2933
13F8	TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD31_DLASTSGA)	32	R/W	0000_0000h	68.3.33/ 2933
13FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD31_BITER_ELINKYES)	16	R/W	0000h	68.3.34/ 2934
13FC	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD31_BITER_ELINKNO)	16	R/W	0000h	68.3.35/ 2935
13FE	TCD Control and Status (DMA_TCD31_CSR)	16	R/W	0000h	68.3.36/ 2936

68.3.5 Control Register (DMA_CR)

The CR defines the basic operating configuration of the DMA. The DMA arbitrates channel service requests in two groups of 16 channels each:

- Group 1 contains channels 31-16
- Group 0 contains channels 15-0

Arbitration within a group can be configured to use either a fixed-priority or a round-robin scheme. For fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities; see the DCHPRIn registers. For round-robin arbitration, the channel priorities are ignored and channels within each group are cycled through (from high to low channel number) without regard to priority.

NOTE

For correct operation, writes to the CR register must be performed only when the DMA channels are inactive; that is, when TCDn_CSR[ACTIVE] bits are cleared.

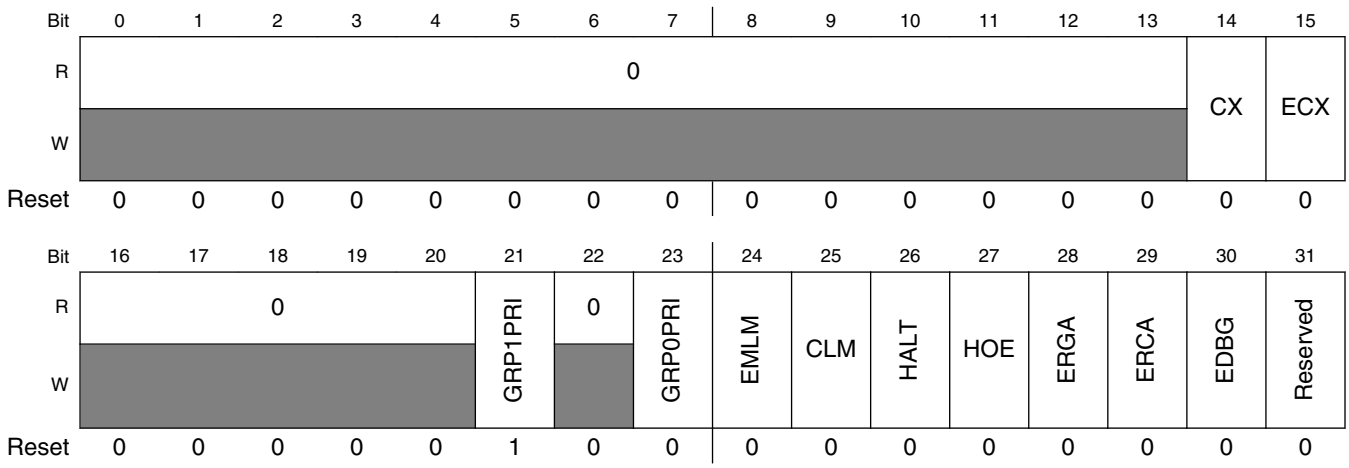
The group priorities operate in a similar fashion. In group fixed priority arbitration mode, channel service requests in the highest priority group are executed first, where priority level 1 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRPnPRI fields of the DMA Control Register (CR). All group priorities must have unique values prior to any channel service requests occurring; otherwise, a configuration error will be reported. For group round robin arbitration, the group priorities are ignored and the groups are cycled through (from high to low group number) without regard to priority.

Minor loop offsets are address offset values added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (MLOFF) is added to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both prior to the addresses being written back into the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (TCDn_SLAST and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

When minor loop mapping is enabled (EMLM is 1), TCDn word2 is redefined. A portion of TCDn word2 is used to specify multiple fields: a source enable bit (SMLOE) to specify the minor loop offset should be applied to the source address (TCDn_SADDR) upon minor loop completion, a destination enable bit (DMLOE) to specify the minor loop offset should be applied to the destination address (TCDn_DADDR) upon minor loop completion, and the sign extended minor loop offset value (MLOFF). The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. When both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

When minor loop mapping is disabled (EMLM is 0), all 32 bits of TCDn word2 are assigned to the NBYTES field.

Address: 0h base + 0h offset = 0h



DMA_CR field descriptions

Field	Description
0–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 CX	Cancel Transfer 0 Normal operation 1 Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CX bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.
15 ECX	Error Cancel Transfer 0 Normal operation 1 Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition, thus updating the Error Status register (DMAx_ES) and generating an optional error interrupt.
16–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 GRP1PRI	Channel Group 1 Priority Group 1 priority level when fixed priority group arbitration is enabled.
22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 GRP0PRI	Channel Group 0 Priority Group 0 priority level when fixed priority group arbitration is enabled.
24 EMLM	Enable Minor Loop Mapping 0 Disabled. TCDn.word2 is defined as a 32-bit NBYTES field. 1 Enabled. TCDn.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled.

Table continues on the next page...

DMA_CR field descriptions (continued)

Field	Description
25 CLM	Continuous Link Mode **NOTE:** Do not use continuous link mode with a channel linking to itself if there is only one minor loop iteration per service request, e.g., if the channel's NBYTES value is the same as either the source or destination size. The same data transfer profile can be achieved by simply increasing the NBYTES value, which provides more efficient, faster processing. 0 A minor loop channel link made to itself goes through channel arbitration before being activated again. 1 A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop.
26 HALT	Halt DMA Operations 0 Normal operation 1 Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
27 HOE	Halt On Error 0 Normal operation 1 Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.
28 ERGA	Enable Round Robin Group Arbitration 0 Fixed priority arbitration is used for selection among the groups. 1 Round robin arbitration is used for selection among the groups.
29 ERCA	Enable Round Robin Channel Arbitration 0 Fixed priority arbitration is used for channel selection within each group. 1 Round robin arbitration is used for channel selection within each group.
30 EDBG	Enable Debug 0 When in debug mode, the DMA continues to operate. 1 When in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system exits debug mode or the EDBG bit is cleared.
31 Reserved	This field is reserved. Reserved

68.3.6 Error Status Register (DMA_ES)

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- A configuration error, that is:
  - An illegal setting in the transfer-control descriptor, or
  - An illegal priority register setting in fixed-arbitration
- An error termination to a bus master read or write cycle

- The UCE bit shows an uncorrectable error occurred while the device has ECC protection on the TCD SRAM
- A cancel transfer with error bit that will be set when a transfer is canceled via the corresponding cancel transfer control bit

See Fault reporting and handling for more details.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0													UCE	ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	0	ERRCHN					SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ES field descriptions

Field	Description
0 VLD	Logical OR of all ERR status bits 0 No ERR bits are set. 1 At least one ERR bit is set indicating a valid error exists that has not been cleared.
1–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 UCE	Uncorrectable ECC error during channel execution. The UCE bit is set when an uncorrectable ECC error occurs on an access generated by the DMA only. If a CPU access to the TCD causes an uncorrectable ECC error, that access will receive a bus error response. **NOTE:** When the eDMA sees a RAM error on an IPS access (when the user is accessing a TCD), the error is reported as a bus abort. When the dma_engine (the execution engine is accessing a TCD) receives a RAM error, it is recorded in the Error Status register, DMA_ES[UCE], along with the channel number. 0 No uncorrectable ECC error 1 The last recorded error was an uncorrectable TCD RAM error
15 ECX	Transfer Canceled 0 No canceled transfers 1 The last recorded entry was a canceled transfer by the error cancel transfer input
16 GPE	Group Priority Error 0 No group priority error 1 The last recorded error was a configuration error among the group priorities. All group priorities are not unique.

Table continues on the next page...

DMA_ES field descriptions (continued)

Field	Description
17 CPE	Channel Priority Error 0 No channel priority error 1 The last recorded error was a configuration error in the channel priorities within a group. Channel priorities within a group are not unique.
18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19–23 ERRCHN	Error Channel Number or Canceled Channel Number The channel number of the last recorded error, excluding GPE and CPE errors, or last recorded error canceled transfer.
24 SAE	Source Address Error 0 No source address configuration error. 1 The last recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE].
25 SOE	Source Offset Error 0 No source offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE].
26 DAE	Destination Address Error 0 No destination address configuration error 1 The last recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE].
27 DOE	Destination Offset Error 0 No destination offset configuration error 1 The last recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE].
28 NCE	NBYTES/CITER Configuration Error 0 No NBYTES/CITER configuration error 1 The last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields. • TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], or • TCDn_CITER[CITER] is equal to zero, or • TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK]
29 SGE	Scatter/Gather Configuration Error 0 No scatter/gather configuration error 1 The last recorded error was a configuration error detected in the TCDn_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. TCDn_DLASTSGA is not on a 32 byte boundary.
30 SBE	Source Bus Error 0 No source bus error 1 The last recorded error was a bus error on a source read

Table continues on the next page...

DMA_ES field descriptions (continued)

Field	Description
31 DBE	Destination Bus Error 0 No destination bus error 1 The last recorded error was a bus error on a destination write

68.3.7 Enable Request Register (DMA_ERQ)

The ERQ register provides a bit map for the 32 channels to enable the request signal for each channel. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. These registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to the ERQ.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ31	ERQ30	ERQ29	ERQ28	ERQ27	ERQ26	ERQ25	ERQ24	ERQ23	ERQ22	ERQ21	ERQ20	ERQ19	ERQ18	ERQ17	ERQ16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ15	ERQ14	ERQ13	ERQ12	ERQ11	ERQ10	ERQ9	ERQ8	ERQ7	ERQ6	ERQ5	ERQ4	ERQ3	ERQ2	ERQ1	ERQ0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERQ field descriptions

Field	Description
0 ERQ31	Enable DMA Request 31 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
1 ERQ30	Enable DMA Request 30

Table continues on the next page...

DMA_ERQ field descriptions (continued)

Field	Description
	0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
2 ERQ29	Enable DMA Request 29 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
3 ERQ28	Enable DMA Request 28 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
4 ERQ27	Enable DMA Request 27 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
5 ERQ26	Enable DMA Request 26 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
6 ERQ25	Enable DMA Request 25 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
7 ERQ24	Enable DMA Request 24 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
8 ERQ23	Enable DMA Request 23 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
9 ERQ22	Enable DMA Request 22 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
10 ERQ21	Enable DMA Request 21 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
11 ERQ20	Enable DMA Request 20 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
12 ERQ19	Enable DMA Request 19 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
13 ERQ18	Enable DMA Request 18

Table continues on the next page...

DMA_ERQ field descriptions (continued)

Field	Description
	0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
14 ERQ17	Enable DMA Request 17 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
15 ERQ16	Enable DMA Request 16 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
16 ERQ15	Enable DMA Request 15 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
17 ERQ14	Enable DMA Request 14 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
18 ERQ13	Enable DMA Request 13 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
19 ERQ12	Enable DMA Request 12 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
20 ERQ11	Enable DMA Request 11 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
21 ERQ10	Enable DMA Request 10 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
22 ERQ9	Enable DMA Request 9 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
23 ERQ8	Enable DMA Request 8 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
24 ERQ7	Enable DMA Request 7 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
25 ERQ6	Enable DMA Request 6

Table continues on the next page...

DMA_ERQ field descriptions (continued)

Field	Description
	0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
26 ERQ5	Enable DMA Request 5 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
27 ERQ4	Enable DMA Request 4 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
28 ERQ3	Enable DMA Request 3 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
29 ERQ2	Enable DMA Request 2 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
30 ERQ1	Enable DMA Request 1 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled
31 ERQ0	Enable DMA Request 0 0 The DMA request signal for the corresponding channel is disabled 1 The DMA request signal for the corresponding channel is enabled

68.3.8 Enable Error Interrupt Register (DMA_EEI)

The EEI register provides a bit map for the 32 channels to enable the error interrupt signal for each channel. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. These registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI register.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EEI31	EEI30	EEI29	EEI28	EEI27	EEI26	EEI25	EEI24	EEI23	EEI22	EEI21	EEI20	EEI19	EEI18	EEI17	EEI16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EEI15	EEI14	EEI13	EEI12	EEI11	EEI10	EEI9	EEI8	EEI7	EEI6	EEI5	EEI4	EEI3	EEI2	EEI1	EEI0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_EEI field descriptions

Field	Description
0 EEI31	Enable Error Interrupt 31 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI30	Enable Error Interrupt 30 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI29	Enable Error Interrupt 29 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI28	Enable Error Interrupt 28 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
4 EEI27	Enable Error Interrupt 27 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI26	Enable Error Interrupt 26 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI25	Enable Error Interrupt 25 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
7 EEI24	Enable Error Interrupt 24 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEI field descriptions (continued)

Field	Description
8 EEI23	Enable Error Interrupt 23 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
9 EEI22	Enable Error Interrupt 22 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
10 EEI21	Enable Error Interrupt 21 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI20	Enable Error Interrupt 20 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
12 EEI19	Enable Error Interrupt 19 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI18	Enable Error Interrupt 18 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI17	Enable Error Interrupt 17 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
15 EEI16	Enable Error Interrupt 16 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
16 EEI15	Enable Error Interrupt 15 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
17 EEI14	Enable Error Interrupt 14 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
18 EEI13	Enable Error Interrupt 13 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
19 EEI12	Enable Error Interrupt 12 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

DMA_EEI field descriptions (continued)

Field	Description
20 EEI11	Enable Error Interrupt 11 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
21 EEI10	Enable Error Interrupt 10 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
22 EEI9	Enable Error Interrupt 9 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
23 EEI8	Enable Error Interrupt 8 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
24 EEI7	Enable Error Interrupt 7 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
25 EEI6	Enable Error Interrupt 6 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
26 EEI5	Enable Error Interrupt 5 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
27 EEI4	Enable Error Interrupt 4 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
28 EEI3	Enable Error Interrupt 3 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
29 EEI2	Enable Error Interrupt 2 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
30 EEI1	Enable Error Interrupt 1 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request
31 EEI0	Enable Error Interrupt 0 0 The error signal for corresponding channel does not generate an error interrupt 1 The assertion of the error signal for corresponding channel generates an error interrupt request

68.3.9 Set Enable Request Register (DMA_SERQ)

The SERQ provides a simple memory-mapped mechanism to set a given bit in the ERQ to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQ to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7
Read	0	0				0		
Write	NOP	SAER	0			SERQ		
Reset	0	0	0	0	0	0	0	0

DMA_SERQ field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 SAER	Set All Enable Requests 0 Set only the ERQ bit specified in the SERQ field 1 Set all bits in ERQ
2 Reserved	This field is reserved.
3–7 SERQ	Set Enable Request Sets the corresponding bit in ERQ.

68.3.10 Clear Enable Request Register (DMA_CERQ)

The CERQ provides a simple memory-mapped mechanism to clear a given bit in the ERQ to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQ to be cleared, disabling all DMA request inputs. If NOP is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 19h offset = 19h

Bit	0	1	2	3	4	5	6	7
Read	0	0				0		
Write	NOP	CAER	0			CERQ		
Reset	0	0	0	0	0	0	0	0

DMA_CERQ field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAER	Clear All Enable Requests 0 Clear only the ERQ bit specified in the CERQ field 1 Clear all bits in ERQ
2 Reserved	This field is reserved.
3–7 CERQ	Clear Enable Request Clears the corresponding bit in ERQ.

68.3.11 Set Enable Error Interrupt Register (DMA_SEEI)

The SEEI provides a simple memory-mapped mechanism to set a given bit in the EEI to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be set. Setting the SAEE bit provides a global set function, forcing the entire EEI contents to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Ah offset = 1Ah

Bit	0	1	2	3	4	5	6	7
Read	0	0				0		
Write	NOP	SAEE	0			SEEI		
Reset	0	0	0	0	0	0	0	0

DMA_SEEI field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 SAEE	Sets All Enable Error Interrupts 0 Set only the EEI bit specified in the SEEI field. 1 Sets all bits in EEI
2 Reserved	This field is reserved.
3–7 SEEI	Set Enable Error Interrupt Sets the corresponding bit in EEI

68.3.12 Clear Enable Error Interrupt Register (DMA_CEEI)

The CEEI provides a simple memory-mapped mechanism to clear a given bit in the EEI to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEI contents to be cleared, disabling all DMA request inputs. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Bh offset = 1Bh

Bit	0	1	2	3	4	5	6	7
Read	0	0				0		
Write	NOP	CAEE	0			CEEI		
Reset	0	0	0	0	0	0	0	0

DMA_CEEI field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAEE	Clear All Enable Error Interrupts 0 Clear only the EEI bit specified in the CEEI field 1 Clear all bits in EEI
2 Reserved	This field is reserved.
3–7 CEEI	Clear Enable Error Interrupt Clears the corresponding bit in EEI

68.3.13 Clear Interrupt Request Register (DMA_CINT)

The CINT provides a simple, memory-mapped mechanism to clear a given bit in the INT to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INT to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INT to be cleared, disabling all DMA interrupt requests. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Ch offset = 1Ch

Bit	0	1	2	3	4	5	6	7
Read	0	0				0		
Write	NOP	CAIR	0			CINT		
Reset	0	0	0	0	0	0	0	0

DMA_CINT field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAIR	Clear All Interrupt Requests 0 Clear only the INT bit specified in the CINT field 1 Clear all bits in INT
2 Reserved	This field is reserved.
3–7 CINT	Clear Interrupt Request Clears the corresponding bit in INT

68.3.14 Clear Error Register (DMA_CERR)

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERR to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERR contents to be cleared, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Dh offset = 1Dh

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	CAEI	0	CERR				
Reset	0	0	0	0	0	0	0	0

DMA_CERR field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CAEI	Clear All Error Indicators 0 Clear only the ERR bit specified in the CERR field 1 Clear all bits in ERR
2 Reserved	This field is reserved.
3–7 CERR	Clear Error Indicator Clears the corresponding bit in ERR

68.3.15 Set START Bit Register (DMA_SSRT)

The SSRT provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Eh offset = 1Eh

Bit	0	1	2	3	4	5	6	7
Read	0	0				0		
Write	NOP	SAST	0			SSRT		
Reset	0	0	0	0	0	0	0	0

DMA_SSRT field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 SAST	Set All START Bits (activates all channels) 0 Set only the TCDn_CSR[START] bit specified in the SSRT field 1 Set all bits in TCDn_CSR[START]
2 Reserved	This field is reserved.
3–7 SSRT	Set START Bit Sets the corresponding bit in TCDn_CSR[START]

68.3.16 Clear DONE Status Bit Register (DMA_CDNE)

The CDNE provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

Address: 0h base + 1Fh offset = 1Fh

Bit	0	1	2	3	4	5	6	7
Read	0	0		0				
Write	NOP	CADN	0	CDNE				
Reset	0	0	0	0	0	0	0	0

DMA_CDNE field descriptions

Field	Description
0 NOP	No Op enable 0 Normal operation 1 No operation, ignore the other bits in this register
1 CADN	Clears All DONE Bits 0 Clears only the TCDn_CSR[DONE] bit specified in the CDNE field 1 Clears all bits in TCDn_CSR[DONE]
2 Reserved	This field is reserved.
3–7 CDNE	Clear DONE Bit Clears the corresponding bit in TCDn_CSR[DONE]

68.3.17 Interrupt Request Register (DMA_INT)

The INT register provides a bit map for the 32 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt on data transfer completion. The outputs of this register are directly routed to the interrupt controller. During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to INT, a 1 in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT register.

Address: 0h base + 24h offset = 24h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_INT field descriptions

Field	Description
0 INT31	Interrupt Request 31

Table continues on the next page...

DMA_INT field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
1 INT30	Interrupt Request 30 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
2 INT29	Interrupt Request 29 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
3 INT28	Interrupt Request 28 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
4 INT27	Interrupt Request 27 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
5 INT26	Interrupt Request 26 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
6 INT25	Interrupt Request 25 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
7 INT24	Interrupt Request 24 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
8 INT23	Interrupt Request 23 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
9 INT22	Interrupt Request 22 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
10 INT21	Interrupt Request 21 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
11 INT20	Interrupt Request 20 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
12 INT19	Interrupt Request 19

Table continues on the next page...

DMA_INT field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
13 INT18	Interrupt Request 18 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
14 INT17	Interrupt Request 17 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
15 INT16	Interrupt Request 16 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
16 INT15	Interrupt Request 15 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
17 INT14	Interrupt Request 14 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
18 INT13	Interrupt Request 13 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
19 INT12	Interrupt Request 12 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
20 INT11	Interrupt Request 11 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
21 INT10	Interrupt Request 10 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
22 INT9	Interrupt Request 9 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
23 INT8	Interrupt Request 8 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
24 INT7	Interrupt Request 7

Table continues on the next page...

DMA_INT field descriptions (continued)

Field	Description
	0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
25 INT6	Interrupt Request 6 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
26 INT5	Interrupt Request 5 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
27 INT4	Interrupt Request 4 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
28 INT3	Interrupt Request 3 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
29 INT2	Interrupt Request 2 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
30 INT1	Interrupt Request 1 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active
31 INT0	Interrupt Request 0 0 The interrupt request for corresponding channel is cleared 1 The interrupt request for corresponding channel is active

68.3.18 Error Register (DMA_ERR)

The ERR provides a bit map for the 32 channels, signaling the presence of an error for each channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI, then logically summed across groups of 16 and 32 channels to form several group error interrupt requests, which are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

Address: 0h base + 2Ch offset = 2Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR31	ERR30	ERR29	ERR28	ERR27	ERR26	ERR25	ERR24	ERR23	ERR22	ERR21	ERR20	ERR19	ERR18	ERR17	ERR16
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_ERR field descriptions

Field	Description
0 ERR31	Error In Channel 31 0 An error in this channel has not occurred 1 An error in this channel has occurred
1 ERR30	Error In Channel 30 0 An error in this channel has not occurred 1 An error in this channel has occurred
2 ERR29	Error In Channel 29 0 An error in this channel has not occurred 1 An error in this channel has occurred
3 ERR28	Error In Channel 28 0 An error in this channel has not occurred 1 An error in this channel has occurred

Table continues on the next page...

DMA_ERR field descriptions (continued)

Field	Description
4 ERR27	Error In Channel 27 0 An error in this channel has not occurred 1 An error in this channel has occurred
5 ERR26	Error In Channel 26 0 An error in this channel has not occurred 1 An error in this channel has occurred
6 ERR25	Error In Channel 25 0 An error in this channel has not occurred 1 An error in this channel has occurred
7 ERR24	Error In Channel 24 0 An error in this channel has not occurred 1 An error in this channel has occurred
8 ERR23	Error In Channel 23 0 An error in this channel has not occurred 1 An error in this channel has occurred
9 ERR22	Error In Channel 22 0 An error in this channel has not occurred 1 An error in this channel has occurred
10 ERR21	Error In Channel 21 0 An error in this channel has not occurred 1 An error in this channel has occurred
11 ERR20	Error In Channel 20 0 An error in this channel has not occurred 1 An error in this channel has occurred
12 ERR19	Error In Channel 19 0 An error in this channel has not occurred 1 An error in this channel has occurred
13 ERR18	Error In Channel 18 0 An error in this channel has not occurred 1 An error in this channel has occurred
14 ERR17	Error In Channel 17 0 An error in this channel has not occurred 1 An error in this channel has occurred
15 ERR16	Error In Channel 16 0 An error in this channel has not occurred 1 An error in this channel has occurred

Table continues on the next page...

DMA_ERR field descriptions (continued)

Field	Description
16 ERR15	Error In Channel 15 0 An error in this channel has not occurred 1 An error in this channel has occurred
17 ERR14	Error In Channel 14 0 An error in this channel has not occurred 1 An error in this channel has occurred
18 ERR13	Error In Channel 13 0 An error in this channel has not occurred 1 An error in this channel has occurred
19 ERR12	Error In Channel 12 0 An error in this channel has not occurred 1 An error in this channel has occurred
20 ERR11	Error In Channel 11 0 An error in this channel has not occurred 1 An error in this channel has occurred
21 ERR10	Error In Channel 10 0 An error in this channel has not occurred 1 An error in this channel has occurred
22 ERR9	Error In Channel 9 0 An error in this channel has not occurred 1 An error in this channel has occurred
23 ERR8	Error In Channel 8 0 An error in this channel has not occurred 1 An error in this channel has occurred
24 ERR7	Error In Channel 7 0 An error in this channel has not occurred 1 An error in this channel has occurred
25 ERR6	Error In Channel 6 0 An error in this channel has not occurred 1 An error in this channel has occurred
26 ERR5	Error In Channel 5 0 An error in this channel has not occurred 1 An error in this channel has occurred
27 ERR4	Error In Channel 4 0 An error in this channel has not occurred 1 An error in this channel has occurred

Table continues on the next page...

DMA_ERR field descriptions (continued)

Field	Description
28 ERR3	Error In Channel 3 0 An error in this channel has not occurred 1 An error in this channel has occurred
29 ERR2	Error In Channel 2 0 An error in this channel has not occurred 1 An error in this channel has occurred
30 ERR1	Error In Channel 1 0 An error in this channel has not occurred 1 An error in this channel has occurred
31 ERR0	Error In Channel 0 0 An error in this channel has not occurred 1 An error in this channel has occurred

68.3.19 Hardware Request Status Register (DMA_HRS)

The HRS register provides a bit map for the DMA channels, signaling the presence of a hardware request for each channel. The hardware request status bits reflect the current state of the register and qualified (via the ERQ fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

NOTE

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ bits.

Address: 0h base + 34h offset = 34h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HRS31	HRS30	HRS29	HRS28	HRS27	HRS26	HRS25	HRS24	HRS23	HRS22	HRS21	HRS20	HRS19	HRS18	HRS17	HRS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HRS15	HRS14	HRS13	HRS12	HRS11	HRS10	HRS9	HRS8	HRS7	HRS6	HRS5	HRS4	HRS3	HRS2	HRS1	HRS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_HRS field descriptions

Field	Description
0 HRS31	Hardware Request Status Channel 31 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 31 is not present 1 A hardware service request for channel 31 is present
1 HRS30	Hardware Request Status Channel 30 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 30 is not present 1 A hardware service request for for channel 30 is present
2 HRS29	Hardware Request Status Channel 29 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 29 is not preset 1 A hardware service request for channel 29 is present
3 HRS28	Hardware Request Status Channel 28 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 28 is not present 1 A hardware service request for channel 28 is present
4 HRS27	Hardware Request Status Channel 27 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	0 A hardware service request for channel 27 is not present 1 A hardware service request for channel 27 is present
5 HRS26	Hardware Request Status Channel 26 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 26 is not present 1 A hardware service request for channel 26 is present
6 HRS25	Hardware Request Status Channel 25 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 25 is not present 1 A hardware service request for channel 25 is present
7 HRS24	Hardware Request Status Channel 24 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 24 is not present 1 A hardware service request for channel 24 is present
8 HRS23	Hardware Request Status Channel 23 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 23 is not present 1 A hardware service request for channel 23 is present
9 HRS22	Hardware Request Status Channel 22 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 22 is not present 1 A hardware service request for channel 22 is present
10 HRS21	Hardware Request Status Channel 21 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 21 is not present 1 A hardware service request for channel 21 is present
11 HRS20	Hardware Request Status Channel 20

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 20 is not present 1 A hardware service request for channel 20 is present
12 HRS19	Hardware Request Status Channel 19 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 19 is not present 1 A hardware service request for channel 19 is present
13 HRS18	Hardware Request Status Channel 18 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 18 is not present 1 A hardware service request for channel 18 is present
14 HRS17	Hardware Request Status Channel 17 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 17 is not present 1 A hardware service request for channel 17 is present
15 HRS16	Hardware Request Status Channel 16 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 16 is not present 1 A hardware service request for channel 16 is present
16 HRS15	Hardware Request Status Channel 15 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 15 is not present 1 A hardware service request for channel 15 is present
17 HRS14	Hardware Request Status Channel 14 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	0 A hardware service request for channel 14 is not present 1 A hardware service request for channel 14 is present
18 HRS13	Hardware Request Status Channel 13 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 13 is not present 1 A hardware service request for channel 13 is present
19 HRS12	Hardware Request Status Channel 12 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 12 is not present 1 A hardware service request for channel 12 is present
20 HRS11	Hardware Request Status Channel 11 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 11 is not present 1 A hardware service request for channel 11 is present
21 HRS10	Hardware Request Status Channel 10 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 10 is not present 1 A hardware service request for channel 10 is present
22 HRS9	Hardware Request Status Channel 9 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 9 is not present 1 A hardware service request for channel 9 is present
23 HRS8	Hardware Request Status Channel 8 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 8 is not present 1 A hardware service request for channel 8 is present
24 HRS7	Hardware Request Status Channel 7

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 7 is not present 1 A hardware service request for channel 7 is present
25 HRS6	Hardware Request Status Channel 6 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 6 is not present 1 A hardware service request for channel 6 is present
26 HRS5	Hardware Request Status Channel 5 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 5 is not present 1 A hardware service request for channel 5 is present
27 HRS4	Hardware Request Status Channel 4 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 4 is not present 1 A hardware service request for channel 4 is present
28 HRS3	Hardware Request Status Channel 3 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 3 is not present 1 A hardware service request for channel 3 is present
29 HRS2	Hardware Request Status Channel 2 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 2 is not present 1 A hardware service request for channel 2 is present
30 HRS1	Hardware Request Status Channel 1 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.

Table continues on the next page...

DMA_HRS field descriptions (continued)

Field	Description
	0 A hardware service request for channel 1 is not present 1 A hardware service request for channel 1 is present
31 HRS0	Hardware Request Status Channel 0 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0 A hardware service request for channel 0 is not present 1 A hardware service request for channel 0 is present

68.3.20 Channel n Priority Register (DMA_DCHPRI*n*)

When fixed-priority channel arbitration is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. Software must program the channel priorities with unique values; otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15. When read, the GRPPRI bits of the DCHPRIn register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRIn registers. The group priority is assigned in the DMA control register.

Address: 0h base + 100h offset + (1d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	ECP	DPA	GRPPRI		CHPRI			
Write								
Reset	0	0	*	*	*	*	*	*

* Notes:
- CHPRI field: See bit field description.
- GRPPRI field: See bit field description.

DMA_DCHPRI*n* field descriptions

Field	Description
0 ECP	Enable Channel Preemption. 0 Channel n cannot be suspended by a higher priority channel's service request. 1 Channel n can be temporarily suspended by the service request of a higher priority channel.

Table continues on the next page...

DMA_DCHPRI*n* field descriptions (continued)

Field	Description
1 DPA	Disable Preempt Ability. 0 Channel n can suspend a lower priority channel. 1 Channel n cannot suspend any channel, regardless of channel priority.
2–3 GRPPRI	Channel n Current Group Priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read-only; writes are ignored. **NOTE:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is, DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] equals 0b1111.
4–7 CHPRI	Channel n Arbitration Priority Channel priority when fixed-priority arbitration is enabled **NOTE:** Reset value for the group and channel priority fields, GRPPRI and CHPRI, is equal to the corresponding channel number for each priority register, that is, DCHPRI31[GRPPRI] = 0b01 and DCHPRI31[CHPRI] = 0b01111.

68.3.21 Channel n Master ID Register (DMA_DCHMID*n*)

The DMA master ID replication registers allow the DMA to use the same protection level and system bus ID of the master programming the DMA's TCD. When enabled, the DMA uses the master ID and protection level stored in the DCHMID register instead of the DMA's default values. When a master, for example, a core, programs a TCD, its master ID and protection level are captured when the TCDn.CSR control attributes are written. Although the scatter/gather operation can change the contents of TCDn_CSR, that operation does not affect the DCHMID *n* registers.

Address: 0h base + 140h offset + (1d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	EMI	PAL	0		MID			
Write								
Reset	0	0	0	0	0	0	0	0

DMA_DCHMID*n* field descriptions

Field	Description
0 EMI	Enable Master ID replication **NOTE:** If Master ID replication is disabled, the privileged protection level (supervisor mode) for DMA transfers is used.

Table continues on the next page...

DMA_DCHMID*n* field descriptions (continued)

Field	Description
	0 Master ID replication is disabled 1 Master ID replication is enabled
1 PAL	Privileged Access Level **NOTE:** The protection level captured in this register reflects the level used when writing the channel's control attributes; lower byte of TCDn.CSR. 0 User protection level for DMA transfers 1 Privileged protection level for DMA transfers
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MID	Master ID DMA's master ID when channel n is active and master ID replication is enabled. **NOTE:** The master ID captured in this register reflects the ID used when writing the channel's control attributes; lower byte of TCDn.CSR.

68.3.22 TCD Source Address (DMA_TCD*n*_SADDR)

Address: 0h base + 1000h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_SADDR field descriptions

Field	Description
0–31 SADDR	Source Address Memory address pointing to the source data.

68.3.23 TCD Transfer Attributes (DMA_TCD*n*_ATTR)

Address: 0h base + 1004h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SMOD					SSIZE			DMOD					DSIZE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_ATTR field descriptions

Field	Description
0–4 SMOD	Source Address Modulo 0 Source address modulo feature is disabled ≠0 This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range.
5–7 SSIZE	Source data transfer size **NOTE:** Using a Reserved value causes a configuration error. 000 8-bit 001 16-bit 010 32-bit 011 64-bit 100 Reserved 101 32-byte burst (4 beats of 64 bits) 110 Reserved 111 Reserved
8–12 DMOD	Destination Address Modulo See the SMOD definition
13–15 DSIZE	Destination data transfer size See the SSIZE definition

68.3.24 TCD Signed Source Address Offset (DMA_TCD*n*_SOFF)

Address: 0h base + 1006h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	SOFF															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_SOFF field descriptions

Field	Description
0–15 SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

68.3.25 TCD Minor Byte Count (Minor Loop Mapping Disabled) (DMA_TCD*n*_NBYTES_MLNO)

This register, or one of the next two registers (TCD_NBYTES_MLOFFNO, TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is disabled (CR[EMLM] = 0)

If minor loop mapping is enabled, see the TCD_NBYTES_MLOFFNO and TCD_NBYTES_MLOFFYES register descriptions for the definition of TCD word 2.

Address: 0h base + 1008h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NBYTES																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_NBYTES_MLNO field descriptions

Field	Description
0–31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. **NOTE:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

68.3.26 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (DMA_TCD*n*_NBYTES_MLOFFNO)

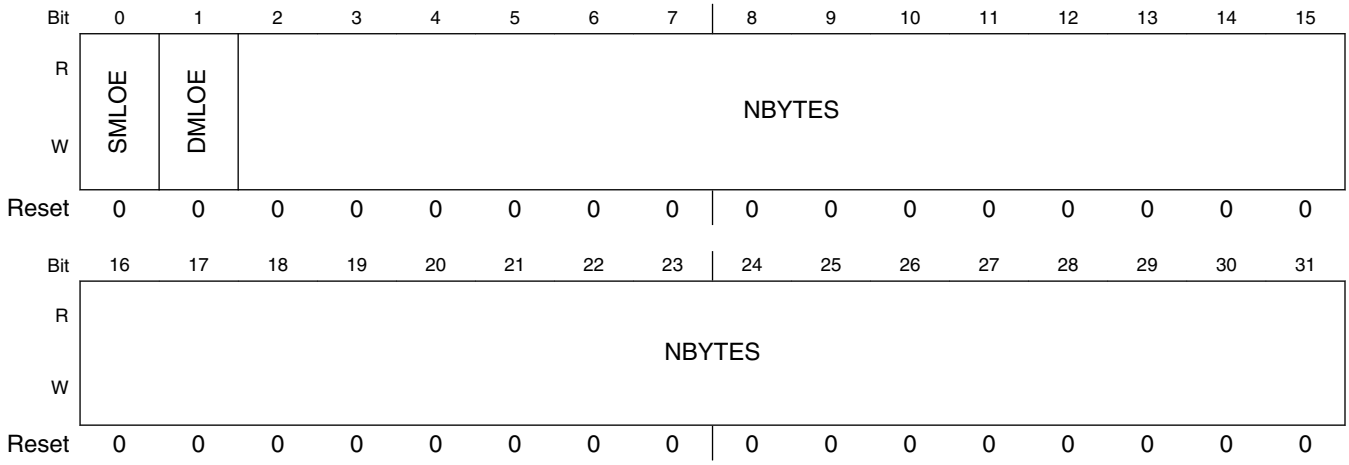
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

If minor loop mapping is enabled and SMLOE or DMLOE is set, then refer to the TCD_NBYTES_MLOFFYES register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

Address: 0h base + 1008h offset + (32d × i), where i=0d to 31d



DMA_TCD*n*_NBYTES_MLOFFNO field descriptions

Field	Description
0 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
1 DMLOE	Destination Minor Loop Offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR
2–31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

68.3.27 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (DMA_TCD*n*_NBYTES_MLOFFYES)

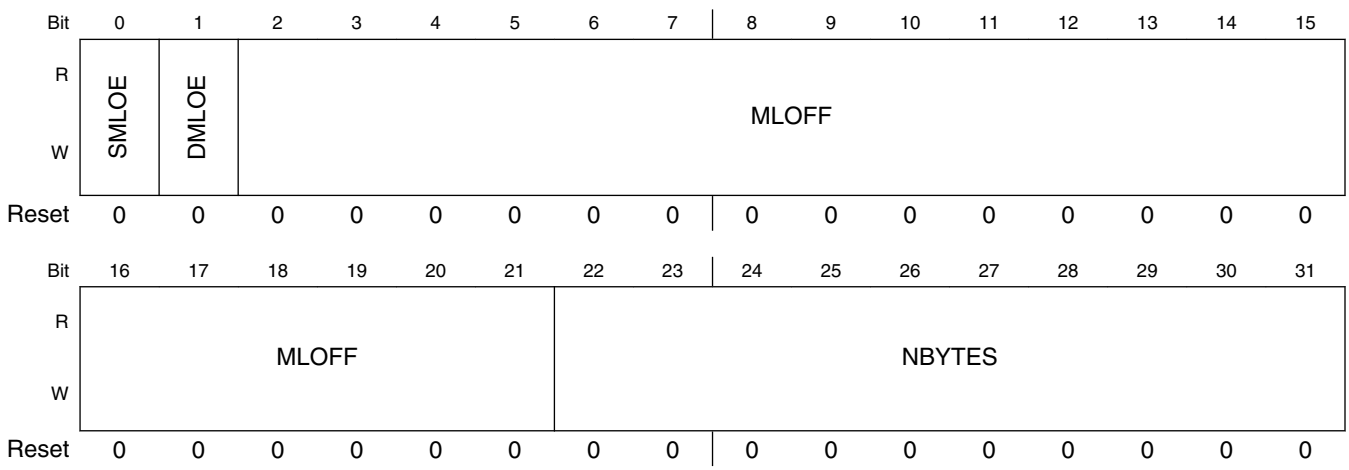
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset is enabled (SMLOE or DMLOE = 1)

If minor loop mapping is enabled and SMLOE and DMLOE are cleared, then refer to the TCD_NBYTES_MLOFFNO register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

Address: 0h base + 1008h offset + (32d × i), where i=0d to 31d



DMA_TCD*n*_NBYTES_MLOFFYES field descriptions

Field	Description
0 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0 The minor loop offset is not applied to the SADDR 1 The minor loop offset is applied to the SADDR
1 DMLOE	Destination Minor Loop Offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0 The minor loop offset is not applied to the DADDR 1 The minor loop offset is applied to the DADDR

Table continues on the next page...

DMA_TCD*n*_NBYTES_MLOFFYES field descriptions (continued)

Field	Description
2–21 MLOFF	If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
22–31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

68.3.28 TCD Last Source Address Adjustment (DMA_TCD*n*_SLAST)

Address: 0h base + 100Ch offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SLAST																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_SLAST field descriptions

Field	Description
0–31 SLAST	Last Source Address Adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure. This register uses two's complement notation; the overflow bit is discarded.

68.3.29 TCD Destination Address (DMA_TCD*n*_DADDR)

Address: 0h base + 1010h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_DADDR field descriptions

Field	Description
0–31 DADDR	Destination Address Memory address pointing to the destination data.

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

If TCDn_CITER[ELINK] is set, the TCDn_CITER register is defined as follows.

Address: 0h base + 1014h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read Write	ELINK	0	LINKCH					CITER
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	CITER							
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_CITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1 Reserved	This field is reserved.
2–6 LINKCH	Minor Loop Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
7–15 CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

68.3.31 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_CITER_ELINKNO)

If TCDn_CITER[ELINK] is cleared, the TCDn_CITER register is defined as follows.

Address: 0h base + 1014h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read Write	ELINK	CITER						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	CITER							
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_CITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–15 CITER	Current Major Iteration Count This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

68.3.32 TCD Signed Destination Address Offset (DMA_TCD*n*_DOFF)

Address: 0h base + 1016h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Read Write	DOFF															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_DOFF field descriptions

Field	Description
0–15 DOFF	Destination Address Signed Offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

68.3.33 TCD Last Destination Address Adjustment/Scatter Gather Address (DMA_TCD*n*_DLASTSGA)

Address: 0h base + 1018h offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DLASTSGA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMA_TCD*n*_DLASTSGA field descriptions

Field	Description
0–31 DLASTSGA	Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather). If (TCDn_CSR[ESG] = 0) then: • Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure. • This field uses two's complement notation for the final destination address adjustment. Otherwise: • This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, otherwise a configuration error is reported.

68.3.34 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (DMA_TCD*n*_BITER_ELINKYES)

If the TCDn_BITER[ELINK] bit is set, the TCDn_BITER register is defined as follows.

Address: 0h base + 101Ch offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	ELINK	0	LINKCH					BITER
Write								
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	BITER							
Write								
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_BITER_ELINKYES field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1 Reserved	This field is reserved.
2–6 LINKCH	Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.
7–15 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

68.3.35 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (DMA_TCD*n*_BITER_ELINKNO)

If the TCDn_BITER[ELINK] bit is cleared, the TCDn_BITER register is defined as follows.

Address: 0h base + 101Ch offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read Write	ELINK	BITER						
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read Write	BITER							
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_BITER_ELINKNO field descriptions

Field	Description
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0 The channel-to-channel linking is disabled 1 The channel-to-channel linking is enabled
1–15 BITER	Starting Major Iteration Count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

68.3.36 TCD Control and Status (DMA_TCD*n*_CSR)

Address: 0h base + 101Eh offset + (32d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7
Read	BWC			MAJORLINKCH				
Write			0					
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	DONE	ACTIVE	MAJORELINK	ESG	DREQ	INTHALF	INTMAJOR	START
Write								
Reset	0	0	0	0	0	0	0	0

DMA_TCD*n*_CSR field descriptions

Field	Description
0–1 BWC	Bandwidth Control Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch. **NOTE:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. 00 No eDMA engine stalls. 01 Enable eDMA master high-priority elevation (HPE) mode. No eDMA engine stalls. 10 eDMA engine stalls for 4 cycles after each R/W. 11 eDMA engine stalls for 8 cycles after each R/W.
2 Reserved	This field is reserved.
3–7 MAJORLINKCH	Major Loop Link Channel Number If (MAJORELINK = 0) then: • No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted. Otherwise: • After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
8 DONE	Channel Done This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero. The software clears it, or the hardware when the channel is activated. **NOTE:** This bit must be cleared to write the MAJORELINK or ESG bits.
9 ACTIVE	Channel Active

Table continues on the next page...

DMA_TCD*n*_CSR field descriptions (continued)

Field	Description
	This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA as the minor loop completes or when any error condition is detected.
10 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. **NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0 The channel-to-channel linking is disabled. 1 The channel-to-channel linking is enabled.
11 ESG	Enable Scatter/Gather Processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory. **NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0 The current channel's TCD is normal format. 1 The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution.
12 DREQ	Disable Request If this flag is set, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches zero. 0 The channel's ERQ bit is not affected. 1 The channel's ERQ bit is cleared when the major loop is complete.
13 INTHALF	Enable an interrupt when major counter is half complete. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes or other types of data movement where the processor needs an early indication of the transfer's progress. **NOTE:** If BITER = 1, do not use INTHALF. Use INTMAJOR instead. 0 The half-point interrupt is disabled. 1 The half-point interrupt is enabled.
14 INTMAJOR	Enable an interrupt when major iteration count completes. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero. 0 The end-of-major loop interrupt is disabled. 1 The end-of-major loop interrupt is enabled.
15 START	Channel Start

Table continues on the next page...

DMA_TCD*n*_CSR field descriptions (continued)

Field	Description
	If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. 0 The channel is not explicitly started. 1 The channel is explicitly started via a software initiated service request.

68.4 Functional description

The operation of the eDMA is described in the following subsections.

68.4.1 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments.

As shown in the following diagram, the first segment involves the channel activation:

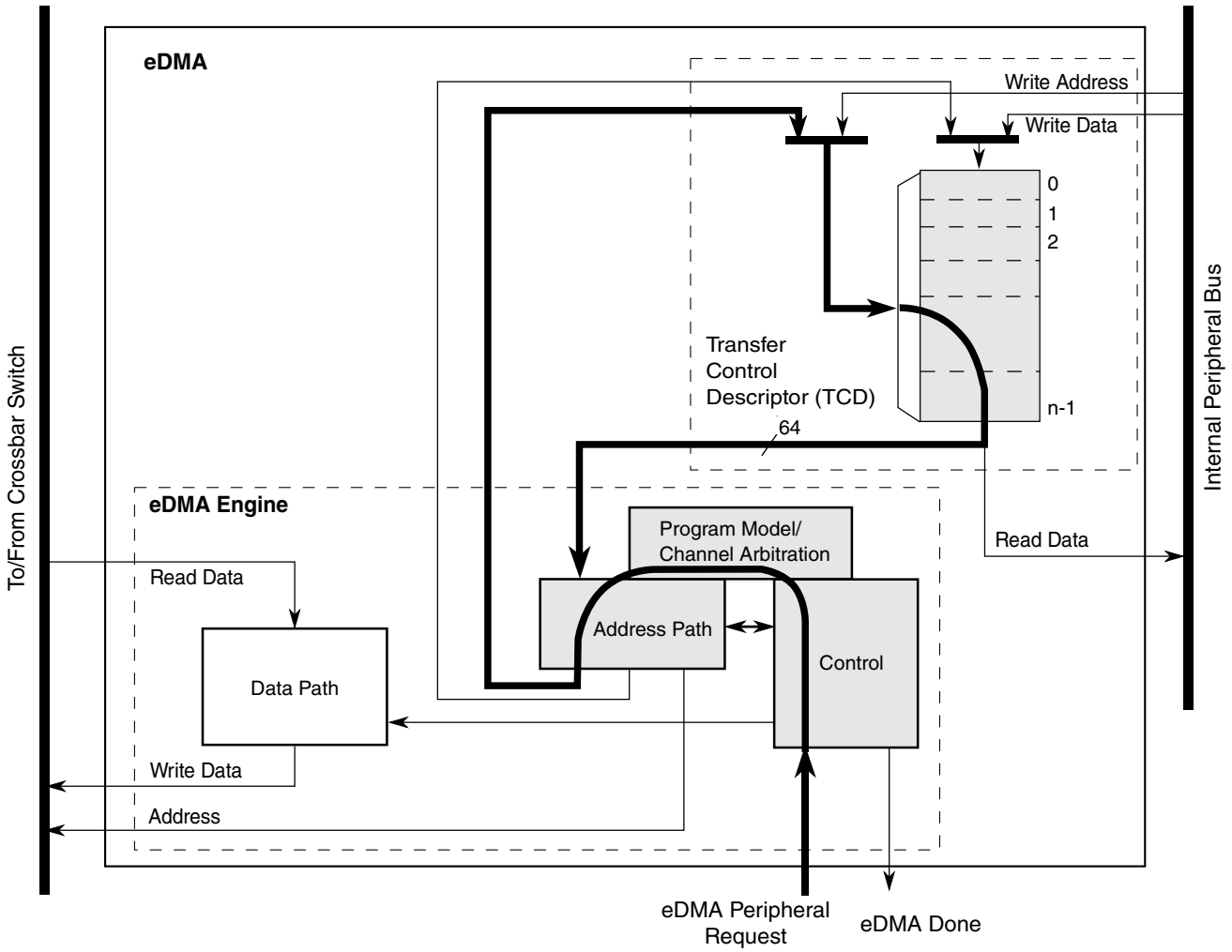


Figure 68-2. eDMA operation, part 1

This example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, the channel arbitration performs, using the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

The following diagram illustrates the second part of the basic data flow:

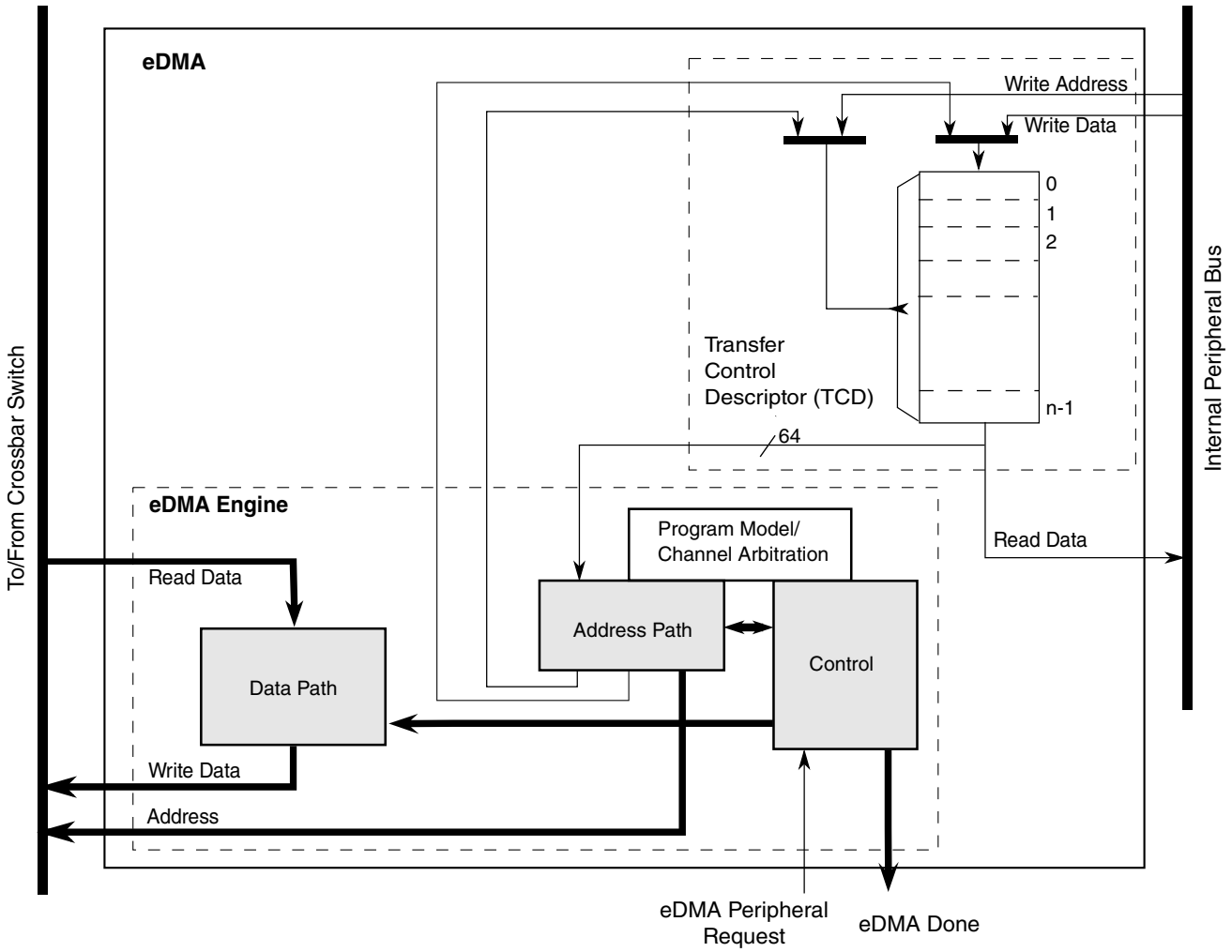


Figure 68-3. eDMA operation, part 2

The modules associated with the data transfer (address path, data path, and control) sequence through the required source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor (if scatter/gather is enabled). The updates to the TCD memory and the assertion of an interrupt request are shown in the following diagram.

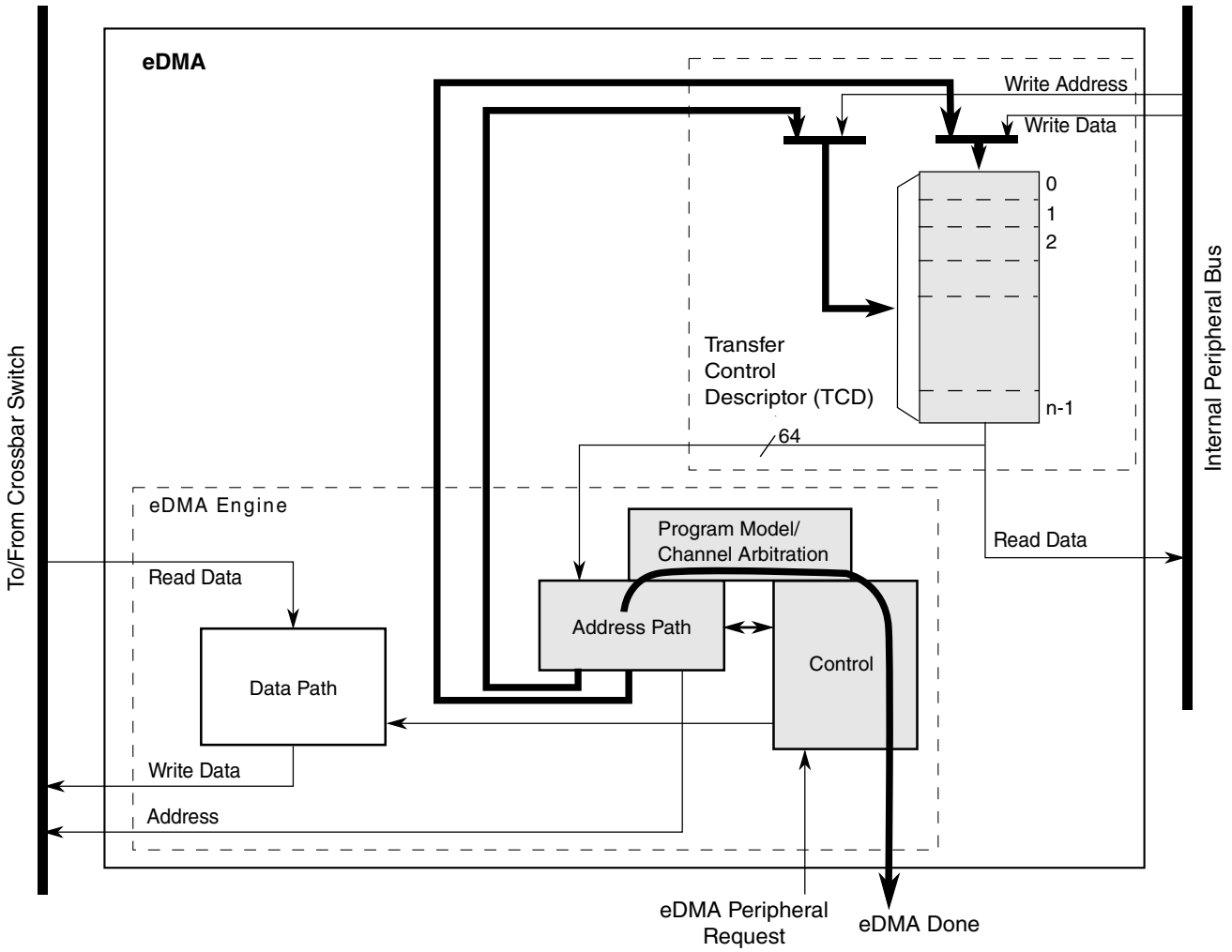


Figure 68-4. eDMA operation, part 3

68.4.2 Fault reporting and handling

Channel errors are reported in the Error Status register (DMAx_ES) and can be caused by:

- A configuration error, which is an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed-Arbitration mode
- An uncorrectable ECC error, or
- An error termination to a bus master read or write cycle

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes are detailed below:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.

- The minor loop byte count must be a multiple of the source and destination transfer sizes.
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.
- In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal. All channel priority levels must be unique when fixed arbitration mode is enabled.

NOTE

When two channels have the same priority, a channel priority error exists and will be reported in the Error Status register. However, the channel number will not be reported in the Error Status register. When all of the channel priorities within a group are not unique, the channel number selected by arbitration is undetermined.

To aid in Channel Priority Error (CPE) debug, set the Halt On Error bit in the DMA's Control Register. If all of the channel priorities within a group are not unique, the DMA will be halted after the CPE error is recorded. The DMA will remain halted and will not process any channel service requests. Once all of the channel priorities are set to unique numbers, the DMA may be enabled again by clearing the Halt bit.

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[E_LINK] bit does not equal the TCDn_BITER[E_LINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error

occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

If an uncorrectable ECC error occurs during a peripheral bus access, the access is terminated with a bus error. The occurrence of an uncorrectable ECC error during an eDMA engine read causes the eDMA engine to stop the active channel immediately. The ECC error and channel number are recorded in the eDMA's Error Status register.

A transfer may be cancelled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the Error Status register (DMAx_ES) is updated with the cancelled channel number and ECX is set. The TCD of a cancelled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD because the aforementioned fields no longer represent the original parameters. When a transfer is cancelled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

NOTE

The cancel transfer request allows the user to stop a large data transfer in the event the full data transfer is no longer needed. The cancel transfer bit does not abort the channel. It simply stops the transferring of data and then retires the channel through its normal shutdown sequence. The application software must handle the context of the cancel. If an interrupt is desired (or not), then the interrupt should be enabled (or disabled) before the cancel request. The application software must clean up the transfer control descriptor since the full transfer did not occur.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has pipeline effect), and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the Error Status register (DMAx_ES). The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request,

are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

68.4.3 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRIn[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption, that is, attempting to preempt a preempting channel, is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected.

A channel's ability to preempt another channel can be disabled by setting DCHPRIn[DPA]. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer, regardless of the lower priority channel's ECP setting. This allows for a pool of low priority, large data-moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority channel from consuming the preempt slot normally available to a true, high priority channel.

68.5 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

68.5.1 eDMA initialization

To initialize the eDMA:

1. Write to the CR if a configuration other than the default is desired.

2. Write the channel priority levels to the DCHPRI*n* registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI register if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.
6. Request channel service via either:
   - Software: setting the TCD*n*_CSR[START]
   - Hardware: slave device asserting its eDMA peripheral request signal

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in the following table, for the selected channel into its internal address path module.

As the TCD is read, the first transfer is initiated on the internal bus, unless a configuration error is detected. Transfers from the source, as defined by TCD*n*_SADDR, to the destination, as defined by TCD*n*_DADDR, continue until the number of bytes specified by TCD*n*_NBYTES are transferred.

When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing executes, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

Table 68-4. TCD Control and Status fields

TCD*n*_CSR field name	Description
START	Control bit to start channel explicitly when using a software initiated DMA service (Automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

The following figure shows how each DMA request initiates one minor-loop transfer, or iteration, without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

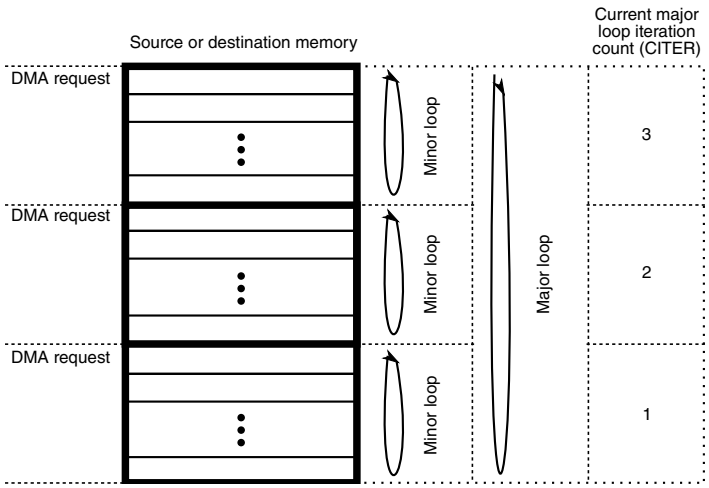


Figure 68-5. Example of multiple loop iterations

The following figure lists the memory array terms and how the TCD settings interrelate.

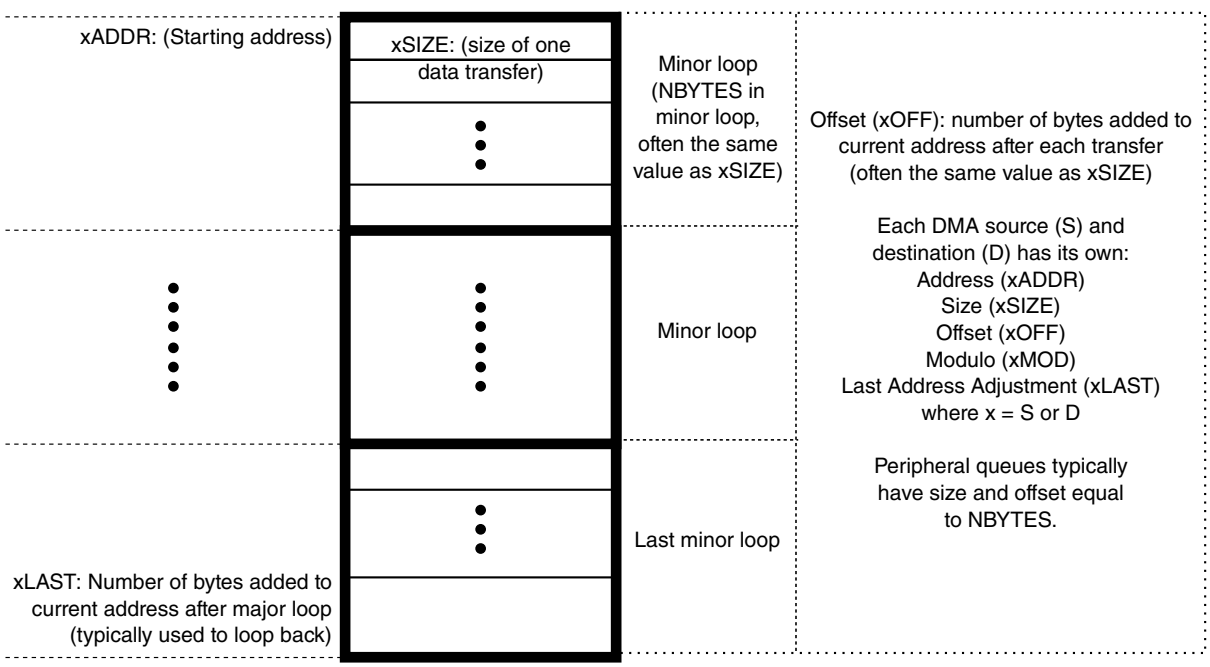


Figure 68-6. Memory array terms

68.5.2 Programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of channel priority error (ES[CPE]).

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the Error Status register (DMAx_ES). If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

Channel priority errors are identified within a group once that group has been selected as the active group. For example:

1. The eDMA is configured for fixed group and fixed channel arbitration modes.
2. Group 1 is the highest priority and all channels are unique in that group.
3. Group 0 is the next highest priority and has two channels with the same priority level.
4. If Group 1 has any service requests, those requests will be executed.
5. After all of Group 1 requests have completed, Group 0 will be the next active group.
6. If Group 0 has a service request, then an undefined channel in Group 0 will be selected and a channel priority error will occur.
7. This repeats until the all of Group 0 requests have been removed or a higher priority Group 1 request comes in.

In this sequence, for item 2, the eDMA acknowledge lines will assert only if the selected channel is requesting service via the eDMA peripheral request signal. If interrupts are enabled for all channels, the user will get an error interrupt, but the channel number for the ERR register and the error interrupt request line may be wrong because they reflect the selected channel. A group priority error is global and any request in any group will cause a group priority error.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest channel/group priority with an active request is selected, but the lowest numbered channel with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

68.5.3 Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

68.5.3.1 Fixed group arbitration, Fixed channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group is selected to execute. If the eDMA is programmed so that the channels within one group use "fixed" priorities, and that group is assigned the highest "fixed" priority of all groups, that group can take all the bandwidth of the eDMA controller. That is, no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

68.5.3.2 Fixed group arbitration, Round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in Fixed group arbitration, Fixed channel arbitration, but all the channels in the highest priority group will be serviced. Service latency will be short on the highest priority group, but could potentially be very much longer as the group priority decreases.

68.5.4 Performing DMA transfers

This section presents examples on how to perform DMA transfers with the eDMA.

68.5.4.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the TCD*n*_CSR[DONE] bit is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a 32-bit port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INT_MAJ] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
    a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
    b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
    c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.

d. Write 32-bits to location 0x2004 → second iteration of the minor loop.

e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.

f. Write 32-bits to location 0x2008 → third iteration of the minor loop.

g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.

h. Write 32-bits to location 0x200C → last iteration of the minor loop → major loop complete.

6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

68.5.4.2 Multiple requests

The following example transfers 32 bytes via two hardware requests, but is otherwise the same as the previous example. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware, that is, eDMA peripheral, request for channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source to destination transfers are executed as follows:

   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.

b. Write 32-bits to location 0x2000 → first iteration of the minor loop.

c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.

d. Write 32-bits to location 0x2004 → second iteration of the minor loop.

e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.

f. Write 32-bits to location 0x2008 → third iteration of the minor loop.

g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.

h. Write 32-bits to location 0x200C → last iteration of the minor loop.

6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0.
8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware, that is, eDMA peripheral, requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads: channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:

a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.

b. Write 32-bits to location 0x2010 → first iteration of the minor loop.

c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.

d. Write 32-bits to location 0x2014 → second iteration of the minor loop.

e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.

f. Write 32-bits to location 0x2018 → third iteration of the minor loop.

g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.

h. Write 32-bits to location 0x201C → last iteration of the minor loop → major loop complete.

14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2 (TCD*n*_BITER).

15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[n] = 1.

16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

68.5.4.3 Using the modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567*x*) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes and the MOD field is set to 4, allowing for a 2^4 byte (16-byte) size queue.

Table 68-5. Modulo example

Transfer Number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

68.5.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

68.5.5.1 Testing for minor loop completion

There are two methods to test for minor loop completion when using software initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD*n*_CSR[START] bit and the TCD*n*_CSR[ACTIVE] bit. The minor-loop-complete condition is indicated by both bits reading zero after the TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using hardware, that is, peripheral, initiated service requests is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware-activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit.

The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

68.5.5.2 Reading the transfer descriptors of active channels

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read while a channel executes. The true values of the SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses, SADDR and DADDR, and NBYTES, which decrement to zero as the transfer progresses, can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

68.5.5.3 Checking channel preemption status

Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in fixed group, fixed channel arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel and/or group priorities are treated as equal, that is, constantly rotating, when Round-Robin Arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

68.5.6 Channel Linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), therefore initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[E_LINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
```

```
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit
2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done→ set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD*n*_CITER[E_LINK] = 1), the TCD*n*_CITER[CITER] field uses a nine bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[E_LINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note

The TCD*n*_CITER[E_LINK] bit and the TCD*n*_BITER[E_LINK] bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel, i.e, use another channel's TCD, at the end of a loop.

Table 68-6. Channel Linking Parameters

Desired Link Behavior	TCD Control Field Name	Description
Link at end of Minor Loop	CITER[E_LINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH]	Link channel number when linking at end of minor loop (current iteration)
Link at end of Major Loop	CSR[MAJOR_E_LINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH]	Link channel number when linking at end of major loop

68.5.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

68.5.7.1 Dynamically changing the channel priority

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode,
2. Disable all the channels, change the channel priorities, then enable the appropriate channels.

68.5.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCD.major.e_link bit during channel execution (see the diagram in TCD structure). This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCD.major.e_link bit at the same time the eDMA engine is retiring the channel. The TCD.major.e_link would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link request.

1. Write 1 to the TCD.major.e_link bit.
2. Read back the TCD.major.e_link bit.
3. Test the TCD.major.e_link request status:
   - If TCD.major.e_link = 1, the dynamic link attempt was successful.
   - If TCD.major.e_link = 0, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCD.major.e_link bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link bit. The TCD.done bit is cleared

automatically by the eDMA engine after a channel begins execution.

68.5.7.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It allows a DMA channel to use multiple TCDs; this enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources.When scatter/gather is enabled and the channel has finished its major loop, a new TCD is fetched from system memory and loaded into that channel's descriptor location in eDMA programmer's model, thus replacing the current descriptor.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCD.e_sg bit at the same time the eDMA engine is retiring the channel. The TCD.e_sg would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the major.linkch field and the e_sg bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD.major.e_link and TCD.e_sg bits to zero on any writes to a channel's TCD.word7 if that channel's TCD.done bit is set indicating the major loop is complete.

NOTE

The user must clear the TCD.done bit before writing the TCD.major.e_link or TCD.e_sg bits. The TCD.done bit is cleared automatically by the eDMA engine after a channel begins execution.

68.5.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request.

When the TCD.major.e_link bit is zero, the TCD.major.linkch field is not used by the eDMA. In this case, the TCD.major.linkch bits may be used for other purposes. This method uses the TCD.major.linkch field as a TCD indentification (ID).

1. When the descriptors are built, write a unique TCD ID in the TCD.major.linkch field for each TCD associated with a channel using dynamic scatter/gather.
2. Write 1b to the TCD.d_req bit.

   Should a dynamic scatter/gather attempt fail, setting the TCD.d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.

3. Write the TCD.dlast_sga field with the scatter/gather address.
4. Write 1b to the TCD.e_sg bit.
5. Read back the 16 bit TCD control/status field.
6. Test the TCD.e_sg request status and TCD.major.linkch value:

   If e_sg = 1b, the dynamic link attempt was successful.

   If e_sg = 0b and the major.linkch (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If e_sg = 0b and the major.linkch (ID) changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

68.5.7.3.2 Method 2 (channel using major loop channel linking)

For a channel using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request. This method uses the TCD.dlast_sga field as a TCD indentification (ID).

1. Write 1b to the TCD.d_req bit.

   Should a dynamic scatter/gather attempt fail, setting the d_req bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offest value.

2. Write theTCD.dlast_sga field with the scatter/gather address.
3. Write 1b to the TCD.e_sg bit.
4. Read back the TCD.e_sg bit.

5. Test the TCD.e_sg request status:

   If e_sg = 1b, the dynamic link attempt was successful.

   If e_sg = 0b, read the 32 bit TCD dlast_sga field.

   If e_sg = 0b and the dlast_sga did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If e_sg = 0b and the dlast_sga changed, the dynamic link attempt was successful (the new TCD's e_sg value cleared the e_sg bit).

Chapter 69
Direct Memory Access Multiplexer

69.1 Chip-specific DMA MUX information

69.1.1 DMA MUX configuration

This device includes two DMA request MUXes that allows up to 32 DMA request signals to be mapped to the first 16 DMA channels [DMA_CH0 - DMA_CH15] for DMACHMUX_0 and the last 16 DMA channels [DMA_CH16 - DMA_CH31] for DMACHMUX_1. There is one DMA Mux module for each 16 channels of DMA as one DMACHMUX is has a configuration of 64 inputs to 16 muxed outputs.

The tables below shows the DMACHMUX_0 and DMACHMUX_1 channel mapping.

Table 69-1. DMA request sources

Source Number	DMACHMUX_0 Source module	DMACHMUX_1 Source module
0	Reserved	Reserved
1	ADC_0	ADC_ 0
2	ADC_1	ADC_1
3	CMP_0	CMP_1
4	BCTU[0]	BCTU[1]
5	FlexCAN_0	FlexCAN_1
6	Reserved	Reserved
7	FlexCAN_2	FlexCAN_3
8	Reserved	Reserved
9	FlexCAN_4	FlexCAN_5
10	Reserved	Reserved
11	Reserved	Reserved
12	SPI_0_TX	SPI_1_TX
13	SPI_0_RX	SPI_1_RX
14	SPI_0_CMD	SPI_1_CMD
15	SPI_2_TX	SPI_3_TX

Table continues on the next page...

Table 69-1. DMA request sources (continued)

Source Number	DMACHMUX_0 Source module	DMACHMUX_1 Source module
16	SPI_2_RX	SPI_3_RX
17	SPI_2_CMD	SPI_3_CMD
18	Reserved	Reserved
19	Reserved	Reserved
20	Reserved	Reserved
21	DSPI_0_TX	DSPI_1_TX
22	DSPI_0_RX	DSPI_1_RX
23	DSPI_0_CMD	DSPI_1_CMD
24	DSPI_0_DSI	DSPI_1_DSI
25	DSPI_2_TX	DSPI_3_TX
26	DSPI_2_RX	DSPI_3_RX
27	DSPI_2_CMD	DSPI_3_CMD
28	DSPI_2_DSI	DSPI_3_DSI
29	LIN_0_TX	LIN_1_TX
30	LIN_0_RX	LIN_1_RX
31	eMIOS_0_chA	eMIOS_0_chE
32	eMIOS_0_chB	eMIOS_0_chF
33	eMIOS_0_chC	eMIOS_0_chG
34	eMIOS_0_chD	eMIOS_0_chH
35	eMIOS_1_chA	eMIOS_1_chE
36	eMIOS_1_chB	eMIOS_1_chF
37	eMIOS_1_chC	eMIOS_1_chG
38	eMIOS_1_chD	eMIOS_1_chH
39	Reserved	I2C_1_TX
40	I2C_0_TX	I2C_1_RX
41	I2C_0_RX	I2C_3_TX
42	I2C_2_TX	I2C_3_RX
43	I2C_2_RX	FlexCAN_7
44	CMP_2	Reserved
45	FLEXCAN_6	Reserved
46	Reserved	Reserved
47	Reserved	Reserved
48	Reserved	Reserved
49	Reserved	SAI_1_TX
50	Reserved	SAI_1_RX
51	SAI_0_TX	Reserved
52	SAI_0_RX	Reserved
53	SAI_2_TX	Reserved
54	SAI_2_RX	Reserved

Table continues on the next page...

Table 69-1. DMA request sources (continued)

Source Number	DMACHMUX_0 Source module	DMACHMUX_1 Source module
55	ENET MAC 0 Timer CH 0	Reserved
56	ENET MAC 0 Timer CH 1	Reserved
57	ENET MAC 0 Timer CH 2	Reserved
58	Reserved	Reserved
59	Reserved	Reserved
60	Reserved	Reserved
61	FCCU DMA request	Reserved
62	Always on	Always on
63	Always on	Always on
Trigger	Trigger 0 - PIT0_CH0	Trigger 2 - PIT0_CH4
	Trigger 1 - PIT0_CH1	Trigger 3 - PIT0_CH5

69.2 Introduction

69.2.1 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 16 DMA channels. This process is illustrated in the following figure.

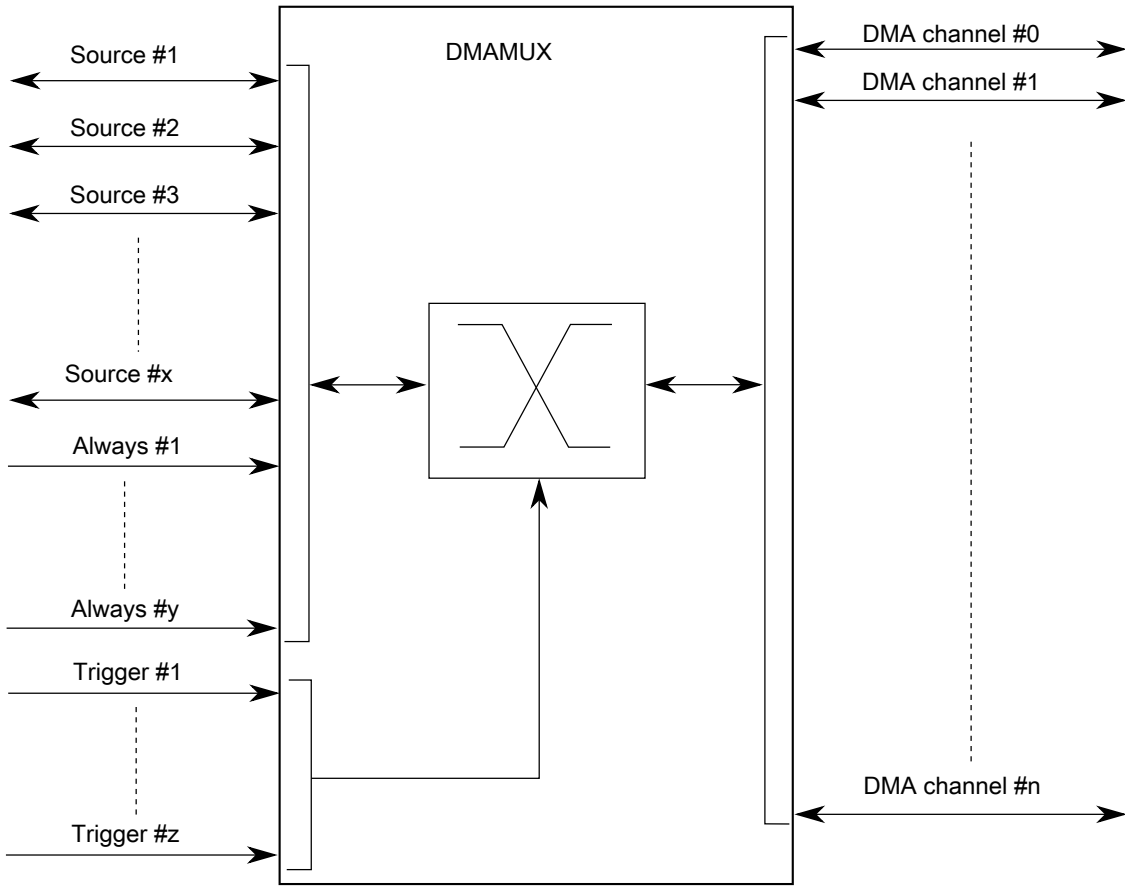


Figure 69-1. DMAMUX block diagram

69.2.2 Features

The DMAMUX module provides these features:

- Up to 61 peripheral slots and up to two always-on slots can be routed to 16 channels.
- 16 independently selectable DMA channel routers.
  - The first two channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of the possible peripheral DMA slots or to one of the always-on slots.

69.2.3 Modes of operation

The following operating modes are available:

- Disabled mode

In this mode, the DMA channel is disabled. Because disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA channel MUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place, for example, changing the period of a DMA trigger.

- Normal mode

  In this mode, a DMA source is routed directly to the specified DMA channel. The operation of the DMAMUX in this mode is completely transparent to the system.

- Periodic Trigger mode

  In this mode, a DMA source may only request a DMA transfer, such as when a transmit buffer becomes empty or a receive buffer becomes full, periodically.

  Configuration of the period is done in the registers of the periodic interrupt timer (PIT). This mode is available only for channels 0–1.

69.3 External signal description

The DMAMUX has no external pins.

69.4 Memory map/register definition

This section provides a detailed description of all memory-mapped registers in the DMAMUX.

DMAMUX memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Channel Configuration register (DMAMUX_CHCFG0)	8	R/W	00h	69.4.1/2966
1	Channel Configuration register (DMAMUX_CHCFG1)	8	R/W	00h	69.4.1/2966
2	Channel Configuration register (DMAMUX_CHCFG2)	8	R/W	00h	69.4.1/2966
3	Channel Configuration register (DMAMUX_CHCFG3)	8	R/W	00h	69.4.1/2966
4	Channel Configuration register (DMAMUX_CHCFG4)	8	R/W	00h	69.4.1/2966
5	Channel Configuration register (DMAMUX_CHCFG5)	8	R/W	00h	69.4.1/2966
6	Channel Configuration register (DMAMUX_CHCFG6)	8	R/W	00h	69.4.1/2966
7	Channel Configuration register (DMAMUX_CHCFG7)	8	R/W	00h	69.4.1/2966
8	Channel Configuration register (DMAMUX_CHCFG8)	8	R/W	00h	69.4.1/2966

Table continues on the next page...

DMAMUX memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
9	Channel Configuration register (DMAMUX_CHCFG9)	8	R/W	00h	69.4.1/2966
A	Channel Configuration register (DMAMUX_CHCFG10)	8	R/W	00h	69.4.1/2966
B	Channel Configuration register (DMAMUX_CHCFG11)	8	R/W	00h	69.4.1/2966
C	Channel Configuration register (DMAMUX_CHCFG12)	8	R/W	00h	69.4.1/2966
D	Channel Configuration register (DMAMUX_CHCFG13)	8	R/W	00h	69.4.1/2966
E	Channel Configuration register (DMAMUX_CHCFG14)	8	R/W	00h	69.4.1/2966
F	Channel Configuration register (DMAMUX_CHCFG15)	8	R/W	00h	69.4.1/2966

69.4.1 Channel Configuration register (DMAMUX_CHCFG*n*)

Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

NOTE

Setting multiple CHCFG registers with the same source value will result in unpredictable behavior. This is true, even if a channel is disabled (ENBL==0).

Before changing the trigger or source settings, a DMA channel must be disabled via CHCFGn[ENBL].

Address: 0h base + 0h offset + (1d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7
Read Write	ENBL	TRIG	SOURCE					
Reset	0	0	0	0	0	0	0	0

DMAMUX_CHCFG*n* field descriptions

Field	Description
0 ENBL	DMA Channel Enable Enables the DMA channel. 0 DMA channel is disabled. This mode is primarily used during configuration of the DMAMux. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel. 1 DMA channel is enabled
1 TRIG	DMA Channel Trigger Enable Enables the periodic trigger capability for the triggered DMA channel.

Table continues on the next page...

DMAMUX_CHCFG*n* field descriptions (continued)

Field	Description
	0 Triggering is disabled. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode) 1 Triggering is enabled. If triggering is enabled and ENBL is set, the DMAMUX is in Periodic Trigger mode.
2–7 SOURCE	DMA Channel Source (Slot) Specifies which DMA source, if any, is routed to a particular DMA channel. See the chip-specific DMAMUX information for details about the peripherals and their slot numbers.

69.5 Functional description

The primary purpose of the DMAMUX is to provide flexibility in the system's use of the available DMA channels.

As such, configuration of the DMAMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in Enabling and configuring sources is followed, the configuration of the DMAMUX may be changed during the normal operation of the system.

Functionally, the DMAMUX channels may be divided into two classes:

- Channels that implement the normal routing functionality plus periodic triggering capability
- Channels that implement only the normal routing functionality

69.5.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 2 channels of the DMAMUX provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames, or packets at fixed intervals without the need for processor intervention.

The trigger is generated by the periodic interrupt timer (PIT); as such, the configuration of the periodic triggering interval is done via configuration registers in the PIT. See the section on periodic interrupt timer for more information on this topic.

Note

Because of the dynamic nature of the system (due to DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

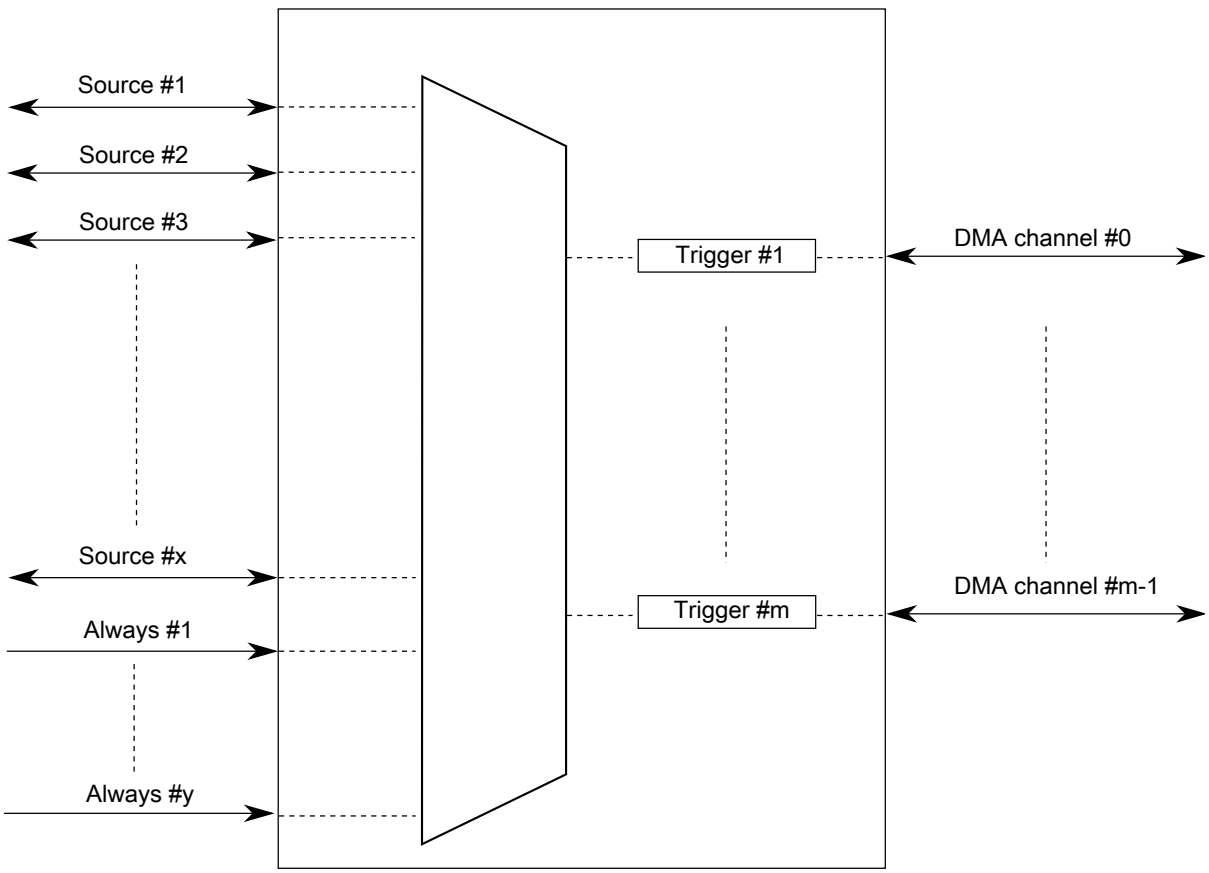


Figure 69-2. DMAMUX triggered channels

The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the peripheral to the DMA until a trigger event has been seen. This is illustrated in the following figure.

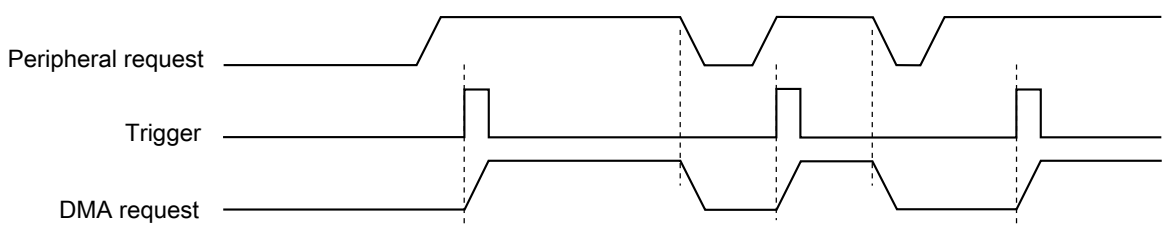


Figure 69-3. DMAMUX channel triggering: normal operation

After the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral reasserts its request and the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, then that trigger will be ignored. This situation is illustrated in the following figure.

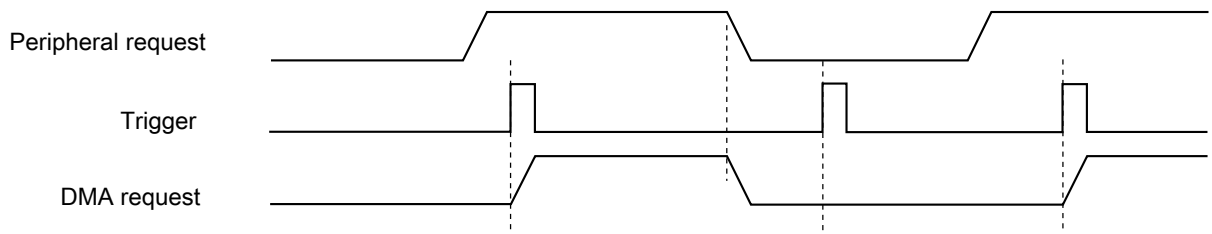


Figure 69-4. DMAMUX channel triggering: ignored trigger

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus

  As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. After it has been set up, the SPI will request DMA transfers, presumably from memory, as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 µs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.

- Using the GPIO ports to drive or sample waveforms

  By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger, including resolution, range of values, and so on, may be found in the periodic interrupt timer section.

69.5.2 DMA channels with no triggering capability

The other channels of the DMAMUX provide the normal routing functionality as described in Modes of operation.

69.5.3 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, there are two additional DMA sources that are always enabled. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the sources that are always enabled provide no such "throttling" of the data transfers. These sources are most useful in the following cases:

- Performing DMA transfers to/from GPIO—Moving data from/to one or more GPIO pins, either unthrottled (that is, as fast as possible), or periodically (using the DMA triggering capability).
- Performing DMA transfers from memory to memory—Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Performing DMA transfers from memory to the external bus, or vice-versa—Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation—Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, an always-enabled DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require that a new start event be sent. This can either be a new software activation, or a transfer request from the DMA channel MUX. The options for doing this are:

- Transfer all data in a single minor loop.

  By configuring the DMA to transfer all of the data in a single minor loop (that is, major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will impose on the system. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use explicit software reactivation.

  In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel by writing to the DMA registers *after every minor loop*. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use an always-enabled DMA source.

In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA channel MUX does the channel reactivation. For this option, the DMA channel should be enabled and pointing to an "always enabled" source. Note that the reactivation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

69.6 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

69.6.1 Reset

The reset state of each individual bit is shown in Memory map/register definition. In summary, after reset, all channels are disabled and must be explicitly enabled before use.

69.6.2 Enabling and configuring sources

To enable a source with periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 2 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Configure the corresponding timer.
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

NOTE

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with periodic triggering capability:

1. Write 0x00 to CHCFG1.

2. Configure channel 1 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC5 to CHCFG1.

The following code example illustrates steps 1 and 4 above:

```
void DMAMUX_Init(uint8_t DMA_CH, uint8_t DMAMUX_SOURCE)
{
    DMAMUX_0.CHCFG[DMA_CH].B.SOURCE = DMAMUX_SOURCE;
    DMAMUX_0.CHCFG[DMA_CH].B.ENBL   = 1;
    DMAMUX_0.CHCFG[DMA_CH].B.TRIG   = 1;
}
```

To enable a source, without periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 2 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is set while CHCFG[TRIG] is cleared.

NOTE

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with no periodic triggering capability:

1. Write 0x00 to CHCFG1.
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG1.

The following code example illustrates steps 1 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
```

```
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00;
*CHCFG1 = 0x85;
```

To disable a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module-specific configuration may be necessary. See the appropriate section for more details.

To switch the source of a DMA channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] bits of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

To switch DMA channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00 to CHCFG8.
3. Write 0x87 to CHCFG8. (In this example, setting CHCFG[TRIG] would have no effect due to the assumption that channel 8 does not support the periodic triggering functionality.)

The following code example illustrates steps 2 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
```

```
In File main.c:
#include "registers.h"
:
:
*CHCFG8 = 0x00;
*CHCFG8 = 0x87;
```

Chapter 70
PRAMC

70.1 Chip-specific PRAMC information

70.1.1 Register reset value

Reset value of PRAMC_PRCR1 register is device specific as shown in the following table

Table 70-1. Register reset value

Register	Register Value
PRAMC_PRCR1	0x0000_0200

70.2 Introduction

This section provides an overview of the Platform RAM Controller. The RAM controller acts as an interface between the system bus (AHB-Lite 2.v6) and the integrated system RAM. It converts the protocols between the system bus and the dedicated RAM array interface.

The RAM controller supports two 64-bit AHB interfaces and a 64-bit RAM array interface. The primary AHB port provides a connection to the platform crossbar for direct RAM access from the various crossbar masters. The backdoor AHB port provides a connection from the flash controller to facilitate calibration overlay access. Shown below in Figure 70-1 is a simplified block diagram depicting the position of the RAM controller within a typical platform architecture.

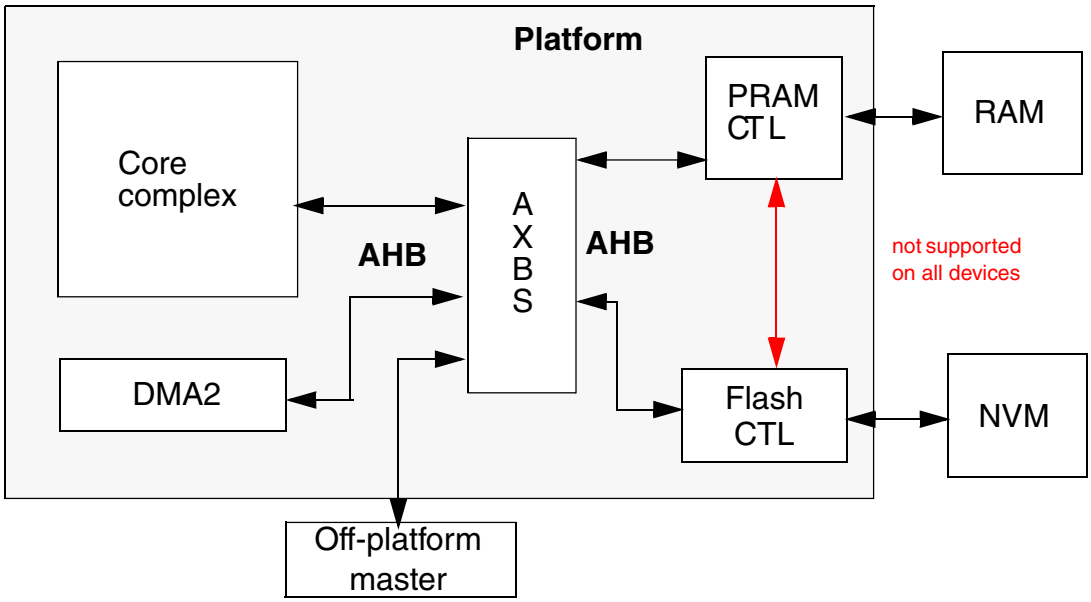


Figure 70-1. Simplified platform block diagram

The following list summarizes the key features of the RAM controller:

- System bus supports 64-bit data + 8-bit ECC AHB interfaces
- Array interface supports a 64-bit data + 8-bit ECC interface
- Late-write support via 64-bit data + 8-bit ECC storage buffer to support single-cycle write accesses
- Configurable read access timing (zero or one wait-state programmable) allowing use in large range of frequency targets
- Read-modify-write operation to support array write size less than a doubleword

The address path of the RAM controller contains a mux that chooses among the addresses presented on the system bus for a read request, either the address from the temporary holding register or the address from the late-write buffer. The temporary holding register contains the address which was presented during the AHB address phase of the write request. The request is stalled from being presented to the RAM by one cycle in order to present simultaneously the address and associated write data, which is not valid until the AHB dataphase. The late-write buffer holds write requests which are delayed to facilitate single-cycle response on the system bus.

The read datapath contains a mux that chooses the source of the read data to be presented on the system bus on a read request. In the event that a read request matches the contents of the late-write buffer, a RAM access is not required and the late-write buffer contents are selected onto the read databus. Otherwise, the read data is a result of a RAM access. Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the RAM controller, including the late-write storage buffer, to support end-to-end ECC (e2eECC) coverage.

The write datapath contains the mux that chooses the source of the write data as well as the control logic for generating read-modify-write operations on writes that are less than 64 bits in size. A write operation can be performed to empty the contents of the late-write buffer. In the case of back-to-back writes, the write may be forced to bypass the late-write buffer and instead be stored directly, precisely in the RAM.

70.3 SRAM controller memory map and register definition

The RAM controller module provides an IPS programming model mapped to a standard on-platform peripheral slot. The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses generate an IPS error termination.

NOTE

Attempted updates to the PRAMC programming model while a PRAMC operation is in progress will result in non-deterministic behavior. Software must be architected to avoid this scenario by ensuring that PRAMC configuration changes are made only during system boot or when only one master is enabled. In multi-core devices, update the PRAMC configuration only when one core is active and no other masters, e.g., DMA or communications modules, are enabled. Move any instruction execution or memory references outside the system RAM while updating the PRAMC configuration, e.g., to the core local memory space.

PRAMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Platform RAM Configuration Register 1 (PRAMC_PRCR1)	32	R/W	0000_0200h	70.3.1/2977

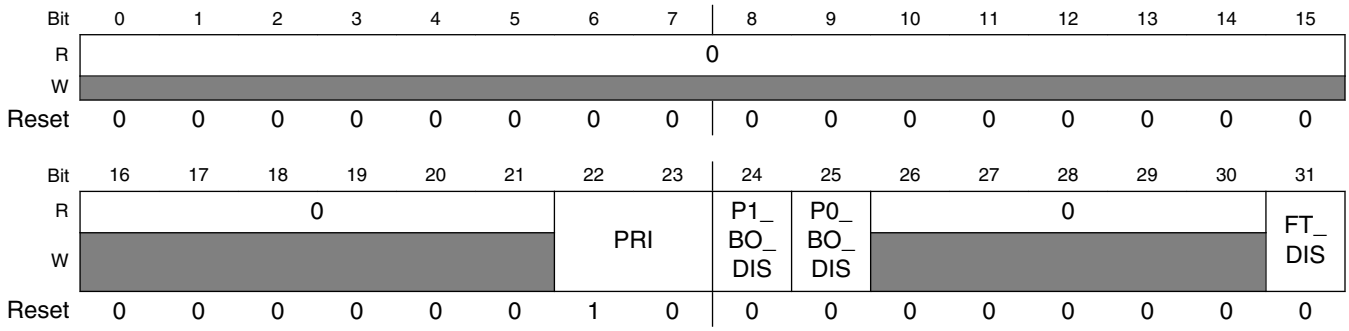
70.3.1 Platform RAM Configuration Register 1 (PRAMC_PRCR1)

The Platform RAM Configuration Register 1 (PRCR1) is used to specify operation of the RAM controller.

NOTE

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

Address: 0h base + 0h offset = 0h



PRAMC_PRCR1 field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 PRI	AHB port arbitration mode. Use this register to select the AHB port arbitration mode. 00 Round Robin arbitration is selected. 01 Port p0 has priority over port p1. 10 Port p1 has priority over port p0. 11 Reserved.
24 P1_BO_DIS	Port p1 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR1[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR1[FT_DIS]=1
25 P0_BO_DIS	Port p0 read burst optimization disable. **NOTE:** The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency. 0 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR1[FT_DIS]=1 1 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR1[FT_DIS]=1
26–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PRAMC_PRCR1 field descriptions (continued)

Field	Description
31 FT_DIS	Flow through disabled. This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset. **NOTE:** Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, e.g., local core memory. 0 RAM read data is passed directly to the system bus, incurring no additional latency 1 RAM read data is registered prior to returning on the system bus, incurring 1 extra cycle of latency

70.4 Functional description

This section describes the functions of the RAM controller.

70.4.1 Read and Write operations

The RAM controller processes read and write requests to on-chip RAM and provides a register interface for access to performance tuning functions.

NOTE

The RAM controller register interface is accessible only in supervisor mode. Accesses in user mode return a transfer error.

70.4.1.1 Reads

Read transfers, of any size, can be configured to complete with a zero or one additional wait state response on the system bus.

70.4.1.2 Optional read wait-state

The RAM controller can be optionally programmed to register RAM read data prior to returning it on the system bus. Setting the PRCRx[FT_DIS] field inserts a register in the read data path for use when operating the controller at high frequencies. The state of PRCRx[FT_DIS] field has no effect on writes.

A random, initial access will take 2 clock cycles (2T) to complete if PRCRx[FT_DIS]=0, and a WRAP4 burst will have an access time of 2-2-2-2. A random, initial access will take 3 clock cycles (3T) to complete if PRCRx[FT_DIS]=1, and a WRAP4 burst will have an access time of 3-2-2-2.

NOTE

The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency.

70.4.2 Writes

This section discusses various write operations of the RAM controller.

70.4.2.1 64-bit writes

Aligned 64-bit writes execute in a single AHB data phase cycle, allowing for zero wait states on back-to-back writes of these sizes. If, during the data phase of a write, a read is requested on the AHB, the write is registered in the late-write buffer, allowing the read to take place without a wait state. The valid buffered or late-write data is stored on the next available array address phase.

Back to back 64-bit writes execute slightly differently whereby the first write bypasses the late-write buffer. Rather, the write data is stored directly to the array in the same cycle in which it is valid on the AHB.

70.4.2.2 Less than 64-bit writes

Writes of size less than 64 bits incur a read-modify-write action as a consequence of the ECC coding scheme's 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The write data is merged into the appropriate byte lanes along with the potentially corrected read data. A new codeword is generated based on the newly formed doubleword. The newly formed doubleword and its

associated checkbits are subsequently written to the RAM. Figure 70-2 provides details on the read-modify-write datapath. For details on the ECC coding scheme, refer to Reliability considerations.

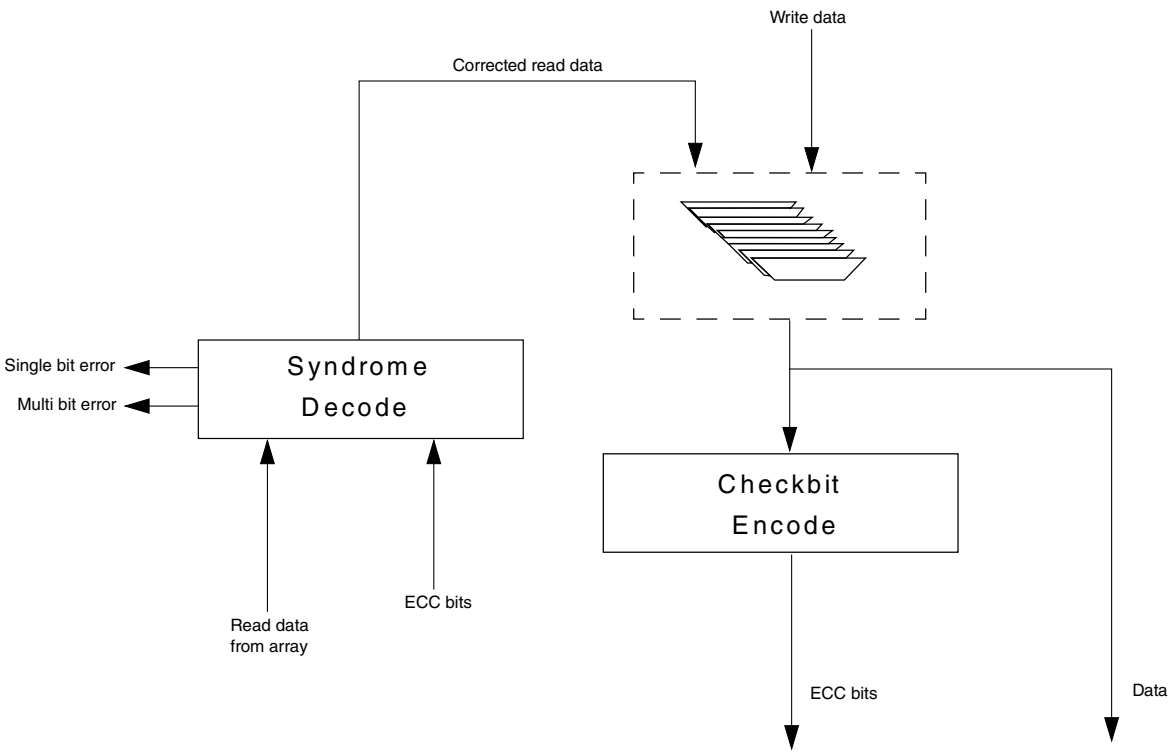


Figure 70-2. Read-modify-write datapath

On a read-modify-write operation, the array read data is registered after it is decoded and before it is merged with the AHB write data. Therefore, writes of size less than 64 bits require the insertion of one wait state before the data phase can be completed.

70.4.2.3 Unaligned writes

The RAM controller is compliant with the AMBA-AHB2.v6 Extensions specification with regard to byte strobes. The size of the transfer is sufficient to cover all bytes being written and covers more bytes in the case of an unaligned transfer. The address of the transfer is rounded down to the nearest boundary of the size of the transaction.

NOTE

Unaligned writes always generate read-modify-write operations in the RAM controller in order to preserve the validity of the ECC codeword.

70.5 Initialization/application information

It is essential that each memory address be written to a known value before it is read, to initialize the ECC. This includes reads generated from the read-modify-write operation which occurs when a write transfer of less than 64 bits or an unaligned write is requested. Without writing an address to a known value first, a read from this address will most likely generate an uncorrectable ECC event.

One possibility for initializing PRAMC memory space is to use a stored 64-bit word instruction such as Store Multiple Word (e_stmw) in Power Architecture VLE.

70.6 Reliability considerations

This section discusses reliability considerations of the RAM controller.

70.6.1 Hsiao ECC algorithm

The e2eECC scheme implements a single-error correction, double-error detection (SECDED) code using the so-called Hsiao odd-weight column criteria. These codes are named for M.Y. Hsiao, an IBM researcher who published extensively in the early 1970s on SECDED codes better suited for implementation in protecting (mainframe) computer memories than traditional Hamming codes.

The Hsiao codes are Hamming distance 4 implementations which provide the SECDED capabilities. The minimum odd-weight constraints defined by Hsiao are relatively simple in the resulting implementation of the parity check H matrix which defines the association between the data (and address) bits and the checkbits. They are:

1. There are no all-zeroes columns.
2. Every column is distinct.
3. Every column contains an odd number of ones, and hence is "odd weight".

In defining the H-matrix for this family of devices, these requirements from Hsiao were applied. Additionally, there are a variety of ECC codeword requirements associated with specific functional requirements associated with the system RAM that further dictated the specific column definitions. In any case, the resulting ECC is organized based on 64 data bits plus 29 address bits (the upper bits of the 32-bit address field minus the 3 bits which select the byte within the 64-bit (8-byte) data field.

The basic H-matrix for this (101, 93) code (93 is the total number of data bits, 101 is the total number of data bits (93) plus 8 checkbits) is shown in the following table. A '*' in the table indicates the corresponding data or address bit is XOR'ed to form the final checkbit value on the left. For 64-bit data writes, the table sections corresponding to D[63:32], D[31:0], and A[31:3] are logically summed (output of each table section is XOR'ed) together to the final value driven on the hwchkbit[7:0] outputs. Note that this table uses *the AHB bit numbering convention where bit[0] is the least significant bit*.

Table 70-2. RAM controller basic H-matrix definition

		XOR sum	Data Bit[1]																																					
			Byte 7								Byte 6								Byte 5								Byte 4							
			63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
Checkbits [7:0]	7		*				*					*			*		*	*		*	*	*	*				*	*				*	*	
	6				*	*			*	*					*		*		*				*				*			*	*			*
	5		*	*	*	*					*				*	*		*		*	*				*	*	*				*			
	4		*		*			*		*					*		*		*			*			*	*						*		
	3										*	*	*	*	*		*				*		*				*	*	*	*					
	2						*		*					*		*			*	*				*	*							*		*
	1			*		*		*		*	*	*	*	*						*		*		*		*						*		*
	0			*			*	*	*				*			*	*	*		*				*				*		*	*		*		
		XOR sum	Byte 3								Byte 2								Byte 1								Byte 0							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Checkbits [7:0]	7						*		*				*		*				*	*			*	*			*				*	*		*
	6		*	*		*	*				*				*				*	*		*	*					*	*				*	*
	5				*	*			*	*	*			*	*	*					*		*	*		*								
	4		*		*	*				*		*	*				*	*	*			*			*	*				*		*		
	3		*	*	*			*						*	*		*	*			*				*		*	*	*			*		
	2			*		*	*	*	*	*	*	*	*			*					*				*		*		*	*		*		
	1			*						*		*			*	*	*			*				*				*			*	*	*	
	0			*		*		*		*				*				*				*				*					*	*		*	*
		XOR sum	Address Bit[2]																																							
			31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Checkbits [7:0]	7			*		*		*		*	*	*		*	*	*		*	*	*		*		*		*	*	*		*		Unused	Unused	Unused
	6			*		*	*	*	*			*	*			*	*	*	*		*		*	*		*	*	*		*				
	5			*	*	*			*	*			*	*		*		*		*	*			*	*	*		*	*	*	*				
	4		*	*	*		*	*		*		*	*	*	*		*					*	*		*	*	*		*	*					
	3		*		*		*	*	*		*		*	*	*			*		*		*	*		*		*	*	*		*				
	2		*	*			*		*	*	*	*				*	*	*	*	*	*	*		*	*		*		*		*				
	1		*		*	*		*	*		*	*		*	*		*		*	*	*		*	*	*					*	*				
	0		*		*	*	*			*	*		*		*	*	*		*		*	*	*			*		*	*		*				

1. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

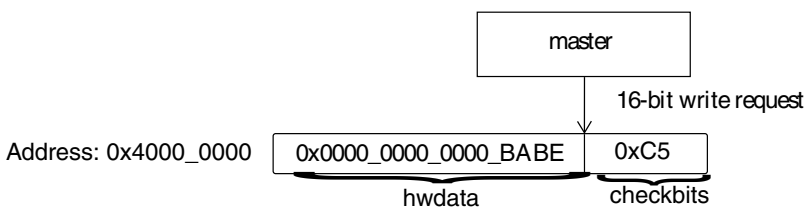
2. Bit numbering is AHB convention: bit 0 is LSB. D[7:0] corresponds to byte at address 0. D[63:56] corresponds to byte at address 7.

70.6.1.1 e2eECC considerations on less-than-64-bit write transactions

Recall writes of size less than 64-bit incur a read-modify-write action as a consequence of the ECC coding scheme 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The RAM controller uses the following checkbit manipulation technique to ensure any faults in the PRAMC control logic associated with performing the read-modify-write action will ultimately be detectable by e2eECC:

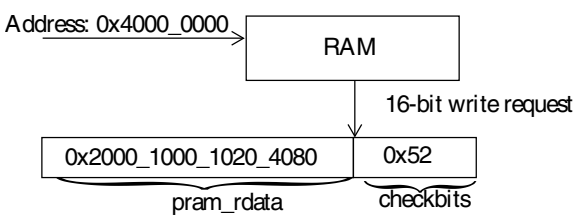
- On an 8-, 16-, or 32-bit write request, the write data is presented by the master on the AHB bus in the correct byte lanes, with the non-pertinent byte lanes zeroed out, and the associated checkbit is based on this "zero-padded" write data.

  Consider the following example whereby the master presents a 16-bit write request at address 0x40000000:

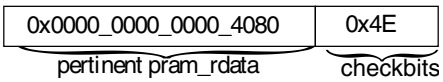


- The RAM controller detects that the request is a less-than-64-bit write request and initiates a read to address 0x40000000. An ECC check and potential single-bit correction is performed on the data returned from the RAM. In addition, the ECC event is reported to the Memory Error Management Unit (MEMU).

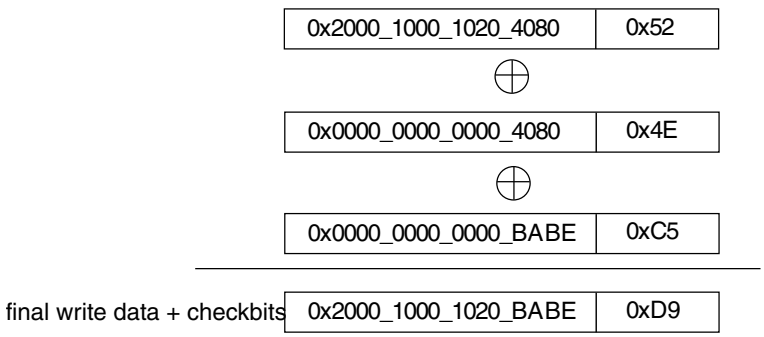
  If a non-correctable ECC event is detected, the AHB request is terminated with error on the system bus. In addition, the ECC event is reported to the MEMU.



- The RAM controller isolates the pertinent byte lanes of the read data by zero-padding the non-pertinent byte lanes and calculating the checkbit contribution associated with the read data in the byte lanes to be updated as a result of the write request.



- Merge the non-pertinent read data bytes with the write data in the appropriate byte lanes. Take the checkbit value that was stored in the RAM and remove the checkbit contribution associated with the read data in the byte lanes to be updated. Then introduce the checkbits associated with the write data. The result is ultimately a checkbit value associated with the complete 64-bit data to be written to the array.



This technique calculates the ultimate checkbit value on a read-modify-write transaction through a series of data manipulations, taking care to avoid discarding the original checkbits provided by the requesting master, such that faults in the crossbar (XBAR) and RAM controller datapath and control logic are covered by e2eECC.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The RAM controller performs EDC after ECC check on all read-modify-write transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the FCCU module.

70.6.2 Transaction monitor

The integrity of the address and data on a system RAM transaction is covered by e2eECC check performed by the requesting master. Fault detection coverage of the address path and control within the RAM controller is handled by a transaction monitor which verifies the integrity of the transactions between the RAM controller and the RAM array. The transaction monitor verifies RAM transactions initiated to service the following types of transactions:

- Direct system bus read or write request
- Read followed by write to fulfill a read-modify write transaction
- Write transaction to empty the late-write buffer

The function of the transaction monitor relies on a feedback path between the RAM controller and RAM array, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated.

This feedback information is used by the RAM controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the RAM controller or the transmission path between the RAM controller and RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

Since writes to the RAM can be buffered, the transaction monitor also tracks the contents of the late-write buffer. When the late-write buffer is emptied, outputs from the RAM are evaluated against the expected contents of the late-write buffer as tracked by the transaction monitor.

As a countermeasure against false misses to the late-write buffer on reads, the transaction monitor also verifies the expected source of data returned on a read as supplied either from the RAM or from the late-write buffer.

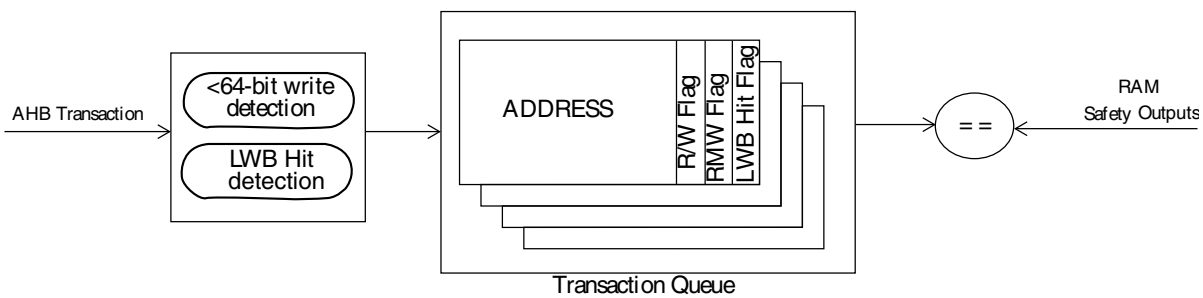


Figure 70-3. RAM controller transaction monitor block diagram

Chapter 71
Flash memory controller

71.1 Chip-specific Flash memory controller information

71.1.1 Master IDs

The registers in the following table contain fields that map to platform bus masters:

Register	Field
Platform Flash Access Protection Register (PFLASH_PFAPR)	M*n*AP
Platform Flash Configuration Register 3 (PFLASH_PFCR3)	P2_M*n*PFE

The "M*n*" portion of each of the field names refers to a bus master by its *logical* master ID. For example, the PFCR1[P0_M15PFE] field refers to the bus master with logic ID 15.

See Table 20-1 for a list of valid bus masters and their logical IDs.

NOTE

The fields associated with bus masters are mapped according to the bus master's logical master ID value, not the AXBS crossbar (XBAR) port number.

71.2 Introduction

The flash memory controller has these functions:

- It acts as an interface between the system bus (AHB-Lite 2.v6) and the flash memory array.
- It serves as the interface to on-chip System RAM that can be used as overlay RAM.

The flash memory controller supports three 64-bit AHB buses and a 256-bit read data interface from each flash memory array. Each AHB port contains a 4-entry, 2-way set-associative mini-cache as well as an associated controller that prefetches sequential lines of data from the flash arrays into the mini-cache. This buffer mechanism serves to deliver flash read data with zero-wait state response on lines that reside in the cache. AHB requests that miss the cache generate the needed flash array access and are forwarded to the AHB upon completion. Each mini-cache entry is 256 bits in size, matching the flash array page size and providing 256 bytes of total buffered storage. Along with the read and write data, the corresponding ECC codewords traverse the entire datapath of the flash memory controller, including the mini-cache, to support end-to-end ECC (e2eECC) coverage.

71.3 Features

The following list summarizes the key features of the flash memory controller:

- Three 64-bit AHB interface (p0, p1, p2) allowing simultaneous access to dedicated prefetch mini-cache per slave port.
- 256-bit read data bus + 64-bit write data bus
- Configurable read buffering and line prefetching support via 4-entry, 2-way set-associative mini-cache plus prefetch controller per AHB port to provide single-cycle "buffer hit" read response.
- Configurable access control based on read/write and AHB master ID attributes.
- Configurable access timing (wait-state programmable) allowing use in a wide range of frequency targets.
- Optional address pipelining capability to maximize throughput.
- Support for reporting of single- and multi-bit flash ECC events on a 64-bit doubleword boundary.
- Configurable overlay remapping of logical flash accesses to on-chip system RAM, on-chip flash memory

71.4 Block diagrams

Figure 71-1 illustrates the connections of the flash memory controller and flash memory array within the chip.

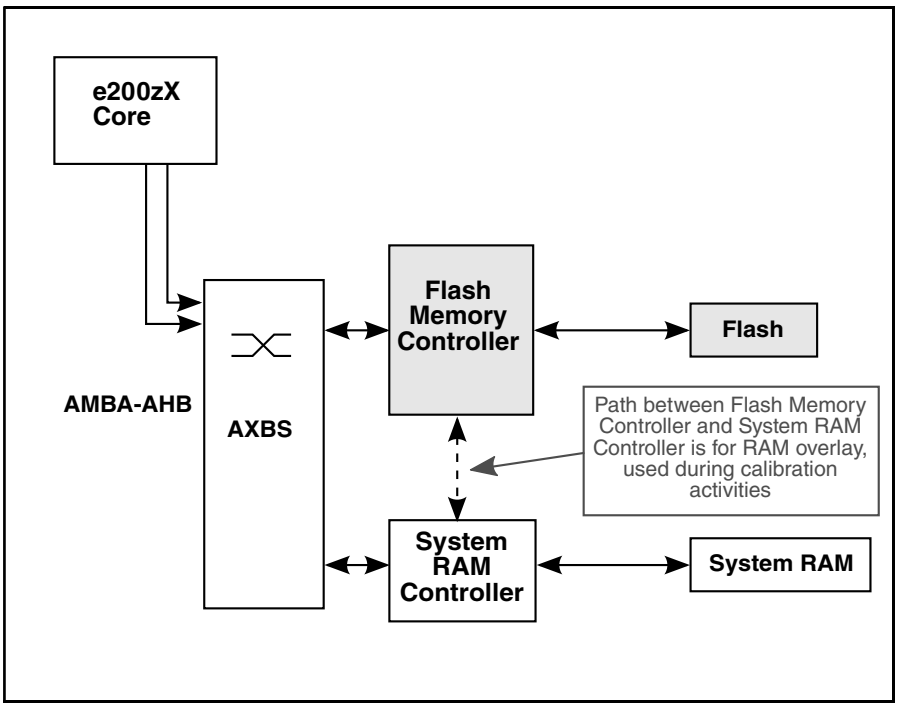


Figure 71-1. Platform-centric simple block diagram with flash memory controller

71.5 Flash memory controller memory map

The flash memory controller module provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model consists of flash memory access configuration and overlay remapping configuration.

The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses or in user mode generates an IPS error termination. The flash controller allows access to the programming model by all system bus masters.

The programming model can only be accessed in supervisor mode.

A write to any flash memory controller register, when executed concurrently with a flash memory access, will result in non-deterministic behavior and may include lockup of the flash memory controller. There is no idle indicator for the flash controller, so software must be architected to prevent any possible conflicts. The following recommendations will help.

- For multi-core devices, start only one core and execute initialization code to completion before starting the remaining cores.

- If flash must be reconfigured during application code execution, temporarily move all required code and data to system RAM before any writes to flash memory controller registers.
- If application software functionality includes invalidating a port read buffer, the recommended method for invalidating a port read buffer is:
  a. Unlock the buffer's enable field by setting the PFLASH_PFCR3[BFEN_LK] field to '0'
  b. Disable the buffer by setting it's associated enable field (PFLASH_PFCR1[Pn_BFEN]) to '0'
  c. If needed, enable the buffer by setting it's associated enable field (PFLASH_PFCR1[Pn_BFEN]) to '1'

The above method requires updating flash memory controller registers, so software must ensure that no flash accesses are issued concurrently with any of the steps. In cases where that cannot be guaranteed, there is a workaround but its use is not generally recommended because it relies on a hardware side effect instead of the flash memory controller's built-in functions intended for invalidating port read buffers. The workaround is to issue an interlock write without an accompanying program or erase operation—an interlock write, without an accompanying program or erase, does not change any flash content.

NOTE

1. If this workaround is implemented in application code, note that an interlock write invalidates only the buffers associated with the system bus port on which the interlock write was issued.
2. See the flash memory chapter for the steps to issue an interlock write.
3. See the flash memory controller's chip-specific information for any restrictions on the interlock write data size.

PFLASH memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Platform Flash Configuration Register 1 (PFLASH_PFCR1)	32	R/W	0000_0601h	71.5.1/2993
4	Platform Flash Configuration Register 2 (PFLASH_PFCR2)	32	R/W	0000_0001h	71.5.2/2996
8	Platform Flash Configuration Register 3 (PFLASH_PFCR3)	32	R/W	0000_0000h	71.5.3/2999
C	Platform Flash Access Protection Register (PFLASH_PFAPR)	32	R/W	FFFF_FFFFh	71.5.4/3001
10	Platform Flash Remap Control Register (PFLASH_PFCRCR)	32	R/W	0000_0000h	71.5.5/3004
14	Platform Flash Remap Descriptor Enable Register (PFLASH_PFCRDE)	32	R/W	0000_0000h	71.5.6/3006

Table continues on the next page...

PFLASH memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
18	Platform Flash Configuration Register 4 (PFLASH_PFCR4)	32	R/W	0000_0001h	71.5.7/3009
100	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD0_Word0)	32	R/W	0000_0000h	71.5.8/3012
104	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD0_Word1)	32	R/W	0000_0000h	71.5.9/3013
108	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD0_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
110	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD1_Word0)	32	R/W	0000_0000h	71.5.8/3012
114	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD1_Word1)	32	R/W	0000_0000h	71.5.9/3013
118	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD1_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
120	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD2_Word0)	32	R/W	0000_0000h	71.5.8/3012
124	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD2_Word1)	32	R/W	0000_0000h	71.5.9/3013
128	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD2_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
130	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD3_Word0)	32	R/W	0000_0000h	71.5.8/3012
134	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD3_Word1)	32	R/W	0000_0000h	71.5.9/3013
138	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD3_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
140	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD4_Word0)	32	R/W	0000_0000h	71.5.8/3012
144	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD4_Word1)	32	R/W	0000_0000h	71.5.9/3013
148	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD4_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
150	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD5_Word0)	32	R/W	0000_0000h	71.5.8/3012
154	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD5_Word1)	32	R/W	0000_0000h	71.5.9/3013
158	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD5_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
160	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD6_Word0)	32	R/W	0000_0000h	71.5.8/3012
164	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD6_Word1)	32	R/W	0000_0000h	71.5.9/3013
168	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD6_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
170	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD7_Word0)	32	R/W	0000_0000h	71.5.8/3012

Table continues on the next page...

PFLASH memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
174	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD7_Word1)	32	R/W	0000_0000h	71.5.9/3013
178	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD7_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
180	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD8_Word0)	32	R/W	0000_0000h	71.5.8/3012
184	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD8_Word1)	32	R/W	0000_0000h	71.5.9/3013
188	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD8_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
190	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD9_Word0)	32	R/W	0000_0000h	71.5.8/3012
194	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD9_Word1)	32	R/W	0000_0000h	71.5.9/3013
198	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD9_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
1A0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD10_Word0)	32	R/W	0000_0000h	71.5.8/3012
1A4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD10_Word1)	32	R/W	0000_0000h	71.5.9/3013
1A8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD10_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
1B0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD11_Word0)	32	R/W	0000_0000h	71.5.8/3012
1B4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD11_Word1)	32	R/W	0000_0000h	71.5.9/3013
1B8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD11_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
1C0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD12_Word0)	32	R/W	0000_0000h	71.5.8/3012
1C4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD12_Word1)	32	R/W	0000_0000h	71.5.9/3013
1C8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD12_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
1D0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD13_Word0)	32	R/W	0000_0000h	71.5.8/3012
1D4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD13_Word1)	32	R/W	0000_0000h	71.5.9/3013
1D8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD13_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
1E0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD14_Word0)	32	R/W	0000_0000h	71.5.8/3012
1E4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD14_Word1)	32	R/W	0000_0000h	71.5.9/3013

Table continues on the next page...

PFLASH memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1E8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD14_Word2)	32	R/W	0000_0000h	71.5.10/ 3014
1F0	Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD15_Word0)	32	R/W	0000_0000h	71.5.8/3012
1F4	Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD15_Word1)	32	R/W	0000_0000h	71.5.9/3013
1F8	Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD15_Word2)	32	R/W	0000_0000h	71.5.10/ 3014

71.5.1 Platform Flash Configuration Register 1 (PFLASH_PFCR1)

The PFlash Configuration Register 1 (PFCR1) controls the operation of Port p0 of the PFLASH_C55FM flash memory controller.

NOTE

See the chip-specific platform flash controller information for information on the masters.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	P0_M15PFE	P0_M14PFE	P0_M13PFE	P0_M12PFE	P0_M11PFE	P0_M10PFE	P0_M9PFE	P0_M8PFE	P0_M7PFE	P0_M6PFE	P0_M5PFE	P0_M4PFE	P0_M3PFE	P0_M2PFE	P0_M1PFE	P0_M0PFE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	APC			RWSC					0	P0_DPFEN	0	P0_IPFEN	0	P0_PFLIM		P0_ BFEN
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1

PFLASH_PFCR1 field descriptions

Field	Description
0 P0_M15PFE	Port0 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
1 P0_M14PFE	Port0 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P0_M13PFE	Port0 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P0_M12PFE	Port0 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P0_M11PFE	Port0 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P0_M10PFE	Port0 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P0_M9PFE	Port0 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P0_M8PFE	Port0 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P0_M7PFE	Port0 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P0_M6PFE	Port0 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P0_M5PFE	Port0 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P0_M4PFE	Port0 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
12 P0_M3PFE	Port0 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P0_M2PFE	Port0 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P0_M1PFE	Port0 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P0_M0PFE	Port0 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–18 APC	Address Pipeline Control. This field controls the number of cycles that a subsequent flash read from the opposite AHB port can be initiated prior to the previous read data being valid. This field applies to the configuration of Port0 and Port1. For valid RWSC and APC combinations please refer to the device data sheet. **NOTE:** 1. A value of '1' in the most significant bit causes the flash controller to function in retrograde/ legacy mode, in which there is no pipelining between the sampling of flash read data and the presentation of the next address for flash lookup. 2. Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the data sheet. 000 Pipelined access to the flash disabled. 001 A pipelined access can be initiated 1 cycle before the previous data is valid 010 A pipelined access can be initiated 2 cycles before the previous data is valid 011 A pipelined access can be initiated 3 cycles before the previous data is valid 1xx Pipelined access to the flash is disabled and one wait state is inserted before a subsequent access can be initiated
19–23 RWSC	Read Wait State Control. This field controls the number of wait-states to be added to the best-case flash array access time for reads. The best-case flash array access time for reads is one cycle. This field must be set to a value corresponding to the operating frequency of the flash memory controller and the actual read access time of the flash memory controller. For valid RWSC and APC combinations please refer to the device data sheet. **NOTE:** 1. Higher operating frequencies require non-zero settings for this field for proper flash operation. 2. Flash operation is not guaranteed for RWSC/APC combinations other than those specified in the device data sheet. This field applies to the configuration of Port0, Port1, and Port2. 00000 No additional wait-states are added 00001 One additional wait-state is added ... 11111 Thirty-one additional wait-states are added

Table continues on the next page...

PFLASH_PFCR1 field descriptions (continued)

Field	Description
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 P0_DPFEN	Port0 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 P0_IPFEN	Port0 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 P0_PFLIM	Port0 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit.
31 P0_BFEN	Port0 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS]=0. After execution of the boot code, as indicated by PFCR3[BAF_DIS]=1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

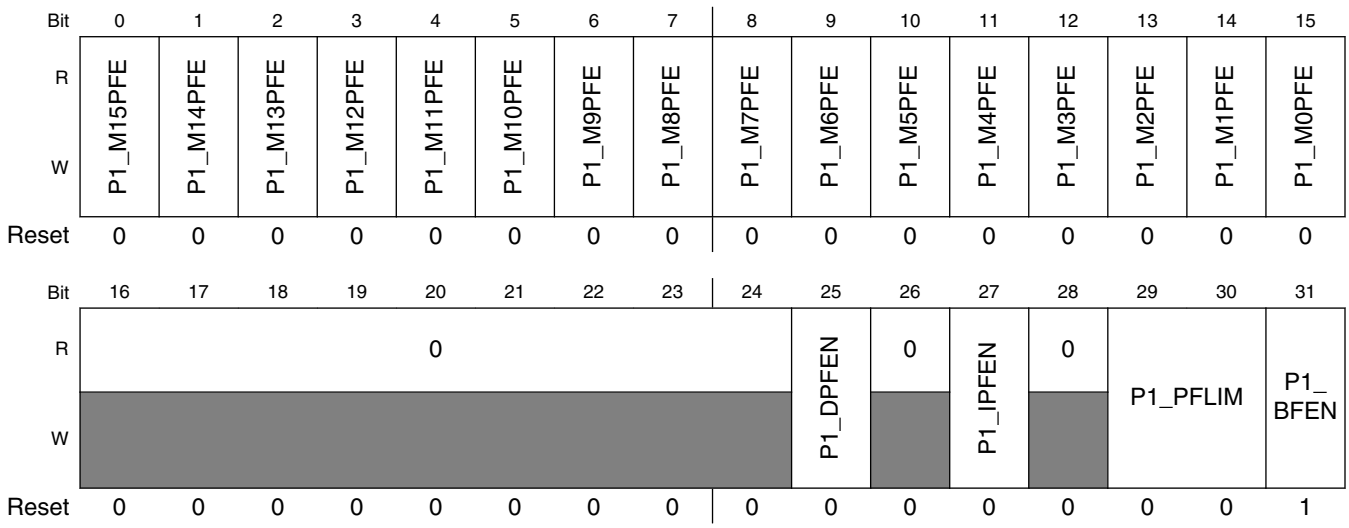
71.5.2 Platform Flash Configuration Register 2 (PFLASH_PFCR2)

The PFlash Configuration Register 2 (PFCR2) controls the operation of Port1 of the PFLASH_C55FM flash memory controller.

NOTE

See the chip-specific platform flash controller information for details about the masters.

Address: 0h base + 4h offset = 4h



PFLASH_PFCR2 field descriptions

Field	Description
0 P1_M15PFE	Port1 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P1_M14PFE	Port1 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P1_M13PFE	Port1 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P1_M12PFE	Port1 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P1_M11PFE	Port1 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P1_M10PFE	Port1 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P1_M9PFE	Port1 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9.

Table continues on the next page...

PFLASH_PFCR2 field descriptions (continued)

Field	Description
	0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P1_M8PFE	Port1 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P1_M7PFE	Port1 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P1_M6PFE	Port1 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P1_M5PFE	Port1 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
11 P1_M4PFE	Port1 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P1_M3PFE	Port1 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P1_M2PFE	Port1 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P1_M1PFE	Port1 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P1_M0PFE	Port1 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 P1_DPFEN	Port1 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 P1_IPFEN	Port1 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access.

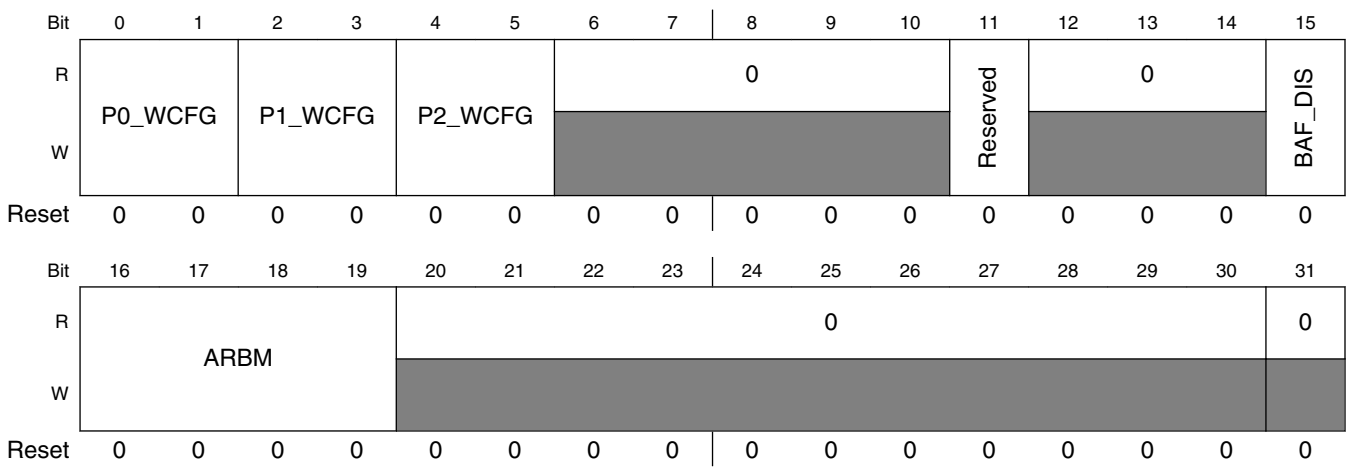
Table continues on the next page...

PFLASH_PFCR2 field descriptions (continued)

Field	Description
	0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 P1_PFLIM	Port1 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit
31 P1_BFEN	Port1 PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS]=0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset. 0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

71.5.3 Platform Flash Configuration Register 3 (PFLASH_PFCR3)

Address: 0h base + 8h offset = 8h



PFLASH_PFCR3 field descriptions

Field	Description
0–1 P0_WCFG	Port0 Way Configuration. This field controls the configuration of the line buffers for a given set across two ways in the controller cache. The indexed set can be organized as a "pool" of available resources, or with a fixed partition between instruction and data buffers.

Table continues on the next page...

PFLASH_PFCR3 field descriptions (continued)

Field	Description
	In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers in an indexed set are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers in an indexed set are partitioned into two groups with way 0 allocated for instruction fetches and way 1 for data accesses. 11 Reserved
2–3 P1_WCFG	Port1 Way Configuration. This field controls the configuration of the line buffers for a given set across two ways in the controller cache. The indexed set can be organized as a "pool" of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers are partitioned into two groups with way 0 allocated for instruction fetches and way 1 for data accesses. 11 Reserved
4–5 P2_WCFG	Port2 Line Buffer Configuration. This field controls the configuration of the line buffers for a given set across both ways in the controller cache. The indexed set can be organized as a “pool” of available resources, or with a fixed partition between instruction and data buffers. In all cases, when a buffer miss occurs, the flash page is assigned a location within the controller cache. Within the indexed set, the way is selected using a least-recently-used replacement policy, and the entry is then marked as most-recently-used for that set. If the flash access is for the next-sequential line (prefetch), the buffer is not marked as most-recently-used until the given address produces a buffer hit. This field is initialized by hardware reset. 00 Both buffers are available for any flash access, that is, there is no partitioning of the buffers based on the access type. 01 Reserved 10 The buffers are partitioned into two groups with buffer 0 allocated for instruction fetches and buffer 1 for data accesses. 11 Reserved
6–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 Reserved	This field is reserved.
12–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PFLASH_PFCR3 field descriptions (continued)

Field	Description
15 BAF_DIS	BAF Disable. This field controls executable access to the BAF (Boot Assist Flash) region of the flash. Once this field is set, attempted instruction accesses targeting the BAF region are aborted and terminated with a system bus error. The affect of this field applies to system bus transfers through both, Port0 and Port1. Data-type accesses to the BAF region are not affected by this field. Once this field is set, it becomes a read-only field and can only be cleared by hardware reset. Once this field is set, any subsequent write attempts to modify this field are ignored with an error-free data transfer termination. This field is initialized by hardware reset. 0 Executable access to the BAF flash region is allowed. 1 Executable access to the BAF flash region is prohibited.
16–19 ARBM	Arbitration Mode. This 4-bit field controls the arbitration of concurrent flash access requests from the three AHB ports of the flash memory controller. In both fixed priority or round-robin modes, write requests are prioritized higher than read requests, and read requests are prioritized higher than speculative prefetch requests whenever multiple ports issue concurrent requests. This field is initialized by hardware reset. **NOTE:** Specifying a reserved value in this field causes it to default to round-robin arbitration. 0000 Fixed priority arbitration with AHB p0>p1>p2 0001 Fixed priority arbitration with AHB p0>p2>p1 0010 Fixed priority arbitration with AHB p2>p0>p1 0011 Fixed priority arbitration with AHB p2>p1>p0 0100 Fixed priority arbitration with AHB p1>p0>p2 0101 Fixed priority arbitration with AHB p1>p2>p0 0110 Reserved 0111 Reserved 1000 Reserved 1001 Reserved 1010 Reserved 1011 Reserved 1100 Reserved 1101 Reserved 1110 Reserved 1111 Round-robin arbitration
20–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

71.5.4 Platform Flash Access Protection Register (PFLASH_PFAPR)

The PFlash Access Protection Register (PFAPR) is used to control read and write accesses to the flash array.

NOTE

See the chip-specific platform flash controller information for details about the actual masters available on the device.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	M0AP		M1AP		M2AP		M3AP		M4AP		M5AP		M6AP		M7AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	M8AP		M9AP		M10AP		M11AP		M12AP		M13AP		M14AP		M15AP	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

PFLASH_PFAPR field descriptions

Field	Description
0–1 M0AP	Master 0 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
2–3 M1AP	Master 1 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
4–5 M2AP	Master 2 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
6–7 M3AP	Master 3 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
8–9 M4AP	Master 4 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table continues on the next page...

PFLASH_PFAPR field descriptions (continued)

Field	Description
10–11 M5AP	Master 5 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
12–13 M6AP	Master 6 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
14–15 M7AP	Master 7 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
16–17 M8AP	Master 8 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
18–19 M9AP	Master 9 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
20–21 M10AP	Master 10 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
22–23 M11AP	Master 11 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

Table continues on the next page...

PFLASH_PFAPR field descriptions (continued)

Field	Description
24–25 M12AP	Master 12 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
26–27 M13AP	Master 13 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
28–29 M14AP	Master 14 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master
30–31 M15AP	Master 15 Access Protection. This field controls whether read and write accesses to the flash are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. 00 No accesses may be performed by this master 01 Only read accesses may be performed by this master 10 Only write accesses may be performed by this master 11 Both read and write accesses may be performed by this master

71.5.5 Platform Flash Remap Control Register (PFLASH_PFCRCR)

The PFCRCR is used to globally enable/disable the calibration remap function.

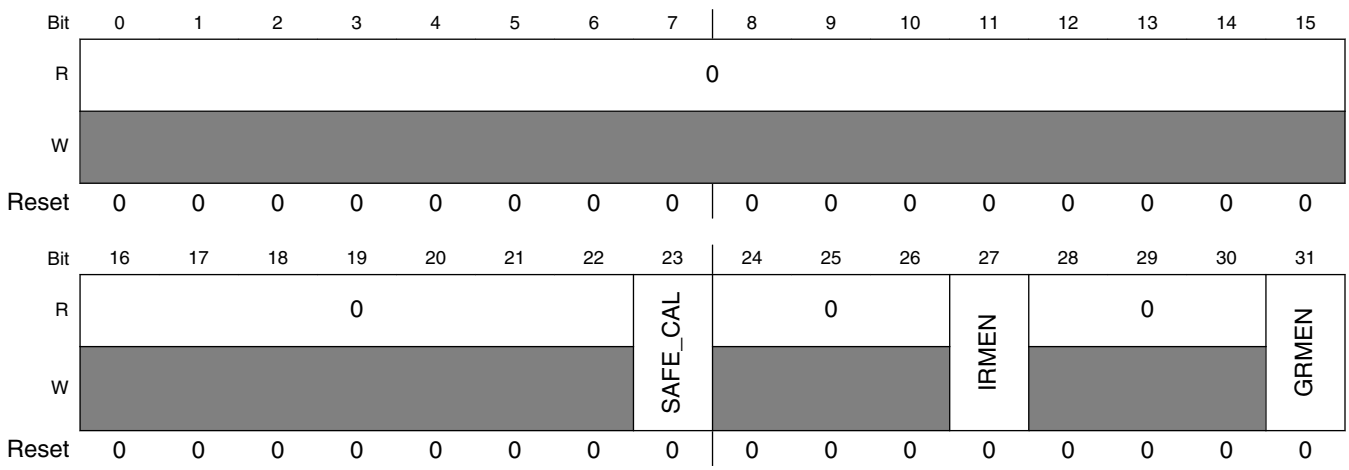
CAUTION

Overlay remap functions, when used with line read buffers enabled (PFLASH_PFCRx[BFy_EN] field(s)='1') have a potential to cause data coherency issues, i.e., prefetched data in a line read buffer can become out-of-sync with data contained at the associated flash address range. To prevent this issue, always clear the flash controller line read buffers prior to enabling any remap region.

To clear the flash controller line read buffers, issue an interlock write command to a valid flash memory address. There will not be an actual flash program operation performed, but the flash controller mini-cache will be cleared as a side effect of executing the command. The interlock write must be issued from a core having appropriate access to flash. The sequence is as follows:

1. Set the C55FMC_MCR[PGM] field to '1'
2. Issue a 64-bit write to a valid flash address on a 64-bit boundary, i.e., the least significant 5 bits of the address are '0'
3. Clear C55FMC_MCR[PGM]

Address: 0h base + 10h offset = 10h



PFLASH_PFCRCR field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 SAFE_CAL	Safe Calibration. This bit facilitates the use of safety-critical calibration data. When this bit is enabled, the flash memory controller is configured to perform redundancy checking on the calibration remap function and reduce the total number of potential calibration regions from 16 to 8. This field is initialized by hardware reset. 1 Established calibration overlay regions are considered safety-critical. 0 Established calibration overlay regions are not considered safety-critical.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 IRMEN	Instruction Remap. This bit enables calibration remapping evaluation on instruction fetches. When PFCRCR[GRMEN] is disabled, this bit is ignored. This field is initialized by hardware reset.

Table continues on the next page...

PFLASH_PFCRCR field descriptions (continued)

Field	Description
	0 Calibration remap evaluation is performed on any incoming data fetch requests only. 1 Calibration remap evaluation is performed on all incoming flash access requests - data and instruction fetches.
28–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 GRMEN	Global Remap Enable. This bit globally enables or disables the calibration remapping evaluation on system flash accesses. This field is initialized by hardware reset. 0 Calibration remap evaluation is not performed on any incoming system flash access requests. 1 Calibration remap evaluation is performed on all incoming system flash access requests.

71.5.6 Platform Flash Remap Descriptor Enable Register (PFLASH_PFCRDE)

The PFlash Calibration Remap Descriptor Enable Register (PFCRDE) is used to enable or disable up to 16 calibration remap descriptors. Note there is also a global remap enable (PFCRCR[GRMEN]) that also must be asserted in conjunction with the individual CRDnEN flags to enable a given remap descriptor.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CRD0EN	CRD1EN	CRD2EN	CRD3EN	CRD4EN	CRD5EN	CRD6EN	CRD7EN	CRD8EN	CRD9EN	CRD10EN	CRD11EN	CRD12EN	CRD13EN	CRD14EN	CRD15EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRDE field descriptions

Field	Description
0 CRD0EN	Calibration Remap Descriptor 0 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset.

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	0 Calibration Remap Descriptor 0 invalid 1 Calibration Remap Descriptor 0 valid
1 CRD1EN	Calibration Remap Descriptor 1 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 1 invalid 1 Calibration Remap Descriptor 1 valid
2 CRD2EN	Calibration Remap Descriptor 2 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 2 invalid 1 Calibration Remap Descriptor 2 valid
3 CRD3EN	Calibration Remap Descriptor 3 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 3 invalid 1 Calibration Remap Descriptor 3 valid
4 CRD4EN	Calibration Remap Descriptor 4 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 4 invalid 1 Calibration Remap Descriptor 4 valid
5 CRD5EN	Calibration Remap Descriptor 5 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 5 invalid 1 Calibration Remap Descriptor 5 valid
6 CRD6EN	Calibration Remap Descriptor 6 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 6 invalid 1 Calibration Remap Descriptor 6 valid
7 CRD7EN	Calibration Remap Descriptor 7 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 7 invalid 1 Calibration Remap Descriptor 7 valid
8 CRD8EN	Calibration Remap Descriptor 8 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 8 invalid 1 Calibration Remap Descriptor 8 valid
9 CRD9EN	Calibration Remap Descriptor 9 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 9 invalid 1 Calibration Remap Descriptor 9 valid
10 CRD10EN	Calibration Remap Descriptor 10 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 10 invalid 1 Calibration Remap Descriptor 10 valid
11 CRD11EN	Calibration Remap Descriptor 11 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 11 invalid 1 Calibration Remap Descriptor 11 valid
12 CRD12EN	Calibration Remap Descriptor 12 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 12 invalid 1 Calibration Remap Descriptor 12 valid
13 CRD13EN	Calibration Remap Descriptor 13 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset.

Table continues on the next page...

PFLASH_PFCRDE field descriptions (continued)

Field	Description
	0 Calibration Remap Descriptor 13 invalid 1 Calibration Remap Descriptor 13 valid
14 CRD14EN	Calibration Remap Descriptor 14 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 14 invalid 1 Calibration Remap Descriptor 14 valid
15 CRD15EN	Calibration Remap Descriptor 15 Enable. This bit indicates whether the corresponding remap descriptor is valid. There is also a global calibration remap enable (PFCRCR[GRMEN]) that also must be asserted to enable any remap functionality. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid. This bit is cleared by power-on reset and unaffected by other types of system reset. 0 Calibration Remap Descriptor 15 invalid 1 Calibration Remap Descriptor 15 valid
16–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

71.5.7 Platform Flash Configuration Register 4 (PFLASH_PFCR4)

The PFlash Configuration Register 4 (PFCR4) specifies operation of Port2 of the flash memory controller.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	P2_M15PFE	P2_M14PFE	P2_M13PFE	P2_M12PFE	P2_M11PFE	P2_M10PFE	P2_M9PFE	P2_M8PFE	P2_M7PFE	P2_M6PFE	P2_M5PFE	P2_M4PFE	P2_M3PFE	P2_M2PFE	P2_M1PFE	P2_M0PFE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0									P2_DPFEN	0	P2_IPFEN	0	P2_PFLIM		P2_BFEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PFLASH_PFCR4 field descriptions

Field	Description
0 P2_M15PFE	Port2 Master 15 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 15. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
1 P2_M14PFE	Port2 Master 14 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 14. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
2 P2_M13PFE	Port2 Master 13 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 13. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
3 P2_M12PFE	Port2 Master 12 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 12. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
4 P2_M11PFE	Port2 Master 11 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 11. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
5 P2_M10PFE	Port2 Master 10 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 10. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
6 P2_M9PFE	Port2 Master 9 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 9. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
7 P2_M8PFE	Port2 Master 8 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 8. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
8 P2_M7PFE	Port2 Master 7 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 7. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
9 P2_M6PFE	Port2 Master 6 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 6. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
10 P2_M5PFE	Port2 Master 5 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 5. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master

Table continues on the next page...

PFLASH_PFCR4 field descriptions (continued)

Field	Description
11 P2_M4PFE	Port2 Master 4 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 4. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
12 P2_M3PFE	Port2 Master 3 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 3. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
13 P2_M2PFE	Port2 Master 2 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 2. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
14 P2_M1PFE	Port2 Master 1 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 1. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
15 P2_M0PFE	Port2 Master 0 Prefetch enable. This bit controls whether prefetching may be triggered by AHB master 0. 0 No prefetching may be triggered by this master 1 Prefetching may be triggered by this master
16–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 P2_DPFEN	Port2 Data Prefetch Enable. This field enables or disables prefetching initiated by a data read access. 0 No prefetching is triggered by a data read access 1 Prefetching may be triggered by any data read access
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 P2_IPFEN	Port2 Instruction Prefetch Enable. This bit enables or disables prefetching initiated by an instruction read access. 0 No prefetching is triggered by an instruction read access 1 Prefetching may be triggered by any instruction read access
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–30 P2_PFLIM	Port2 PFlash Prefetch Limit. This field controls the prefetch algorithm used by the prefetch controller. This field defines a limit on the maximum number of sequential prefetches which will be attempted between buffer misses. In all situations when enabled, only a single prefetch is initiated on each buffer miss or hit. 00 No prefetching or buffering is performed. 01 The referenced line is prefetched on a buffer miss, that is, prefetch on miss. 10 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit. 11 The referenced line is prefetched on a buffer miss, or the next sequential line is prefetched on a buffer hit (if not already present), that is, prefetch on miss or hit
31 P2_BFEN	PFlash Line Read Buffers Enable. This bit enables or disables line read buffer hits. It is also used to invalidate the buffers. This bit can only be updated while PFCR3[BAF_DIS]=0. After execution of the boot code, as indicated by PFCR3[BAF_DIS] = 1, this bit becomes read-only until the next hardware reset.

Table continues on the next page...

PFLASH_PFCR4 field descriptions (continued)

Field	Description
	0 The line read buffers are disabled from satisfying read requests, and all buffer valid bits are cleared. 1 The line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.

71.5.8 Platform Flash Calibration Region Descriptor n Word0 (PFLASH_PFCRD*n*_Word0)

Each 96-bit (12-byte) region descriptor specifies an overlay region where a flash access can be remapped during calibration and debug. The calibration remap descriptors are organized sequentially as 128-bit (16-byte) structures in the Platform Flash Controller's programming model. Each of the three 32-bit words that define a single calibration region are detailed in the subsequent sections; the fourth word is unused.

The first word of the flash memory controller overlay region descriptor defines the 0-modulo-size logical start (byte) address of the calibration remap region. It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

The hardware performs basic checks on the validity of the logical start address when this word is written; if a non-supported logical start address is defined, the register write is terminated with an error and the register left unchanged and the valid bit cleared. Successful writes to this word also clear the calibration remap descriptor's valid bit.

Address: 0h base + 100h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	LSTARTADDR																												0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRD*n*_Word0 field descriptions

Field	Description
0–27 LSTARTADDR	Logical Start Address. This field defines the most significant bits of the 0-modulo-size logical start address of the overlay remap region. It corresponds to the logical system address which maps to the flash memory space. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

71.5.9 Platform Flash Calibration Region Descriptor n Word1 (PFLASH_PFCRD*n*_Word1)

The second word of the flash memory controller calibration region descriptor defines the 0-modulo-size byte physical start (byte) address of the calibration remap region.

The contents of this word define the targeted destination overlay memory. It is software's responsibility to guarantee the low-order bits of the address, as defined by the remap descriptor size, are zeroed to enable the remap descriptor hit logic to function correctly.

Attempted writes to this register with illegal values, as defined by the sizes and locations of the overlay memories, are terminated with an error and clear the valid bit. Successful writes to this word also clear the calibration remap descriptor's valid bit.

Address: 0h base + 104h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	PSTARTADDR																												0			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRD*n*_Word1 field descriptions

Field	Description
0–27 PSTARTADDR	Calibration Remap Descriptor *n* Physical Start Address - This field defines the most significant bits of the 0-modulo-size physical start byte address of the calibration remap descriptor. This address corresponds to the physical address which maps to the destination calibration overlay memory. Any write to PFCRDn.Word{0,1,2} clears the corresponding PFCRDE[CRDnEN] bit, leaving the calibration remap descriptor invalid.
28–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

71.5.10 Platform Flash Calibration Region Descriptor n Word2 (PFLASH_PFCRD*n*_Word2)

The third word of the calibration region descriptor defines a per-master calibration remap enable and the remap region size. For cacheable spaces being remapped, the minimum region size is 32 bytes to match the flash page and cache line sizes. Writes to this word clear the calibration remap descriptor's valid bit.

Address: 0h base + 108h offset + (16d × i), where i=0d to 15d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	M0EN	M1EN	M2EN	M3EN	M4EN	M5EN	M6EN	M7EN	M8EN	M9EN	M10EN	M11EN	M12EN	M13EN	M14EN	M15EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0											CRDSize				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PFLASH_PFCRD*n*_Word2 field descriptions

Field	Description
0 M0EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 0 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 0 as defined by this region descriptor.
1 M1EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 1 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 1 as defined by this region descriptor.
2 M2EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 2 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 2 as defined by this region descriptor.
3 M3EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 3 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 3 as defined by this region descriptor.
4 M4EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 4 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 4 as defined by this region descriptor.
5 M5EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 5 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 5 as defined by this region descriptor.
6 M6EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 6 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 6 as defined by this region descriptor.
7 M7EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 7 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 7 as defined by this region descriptor.
8 M8EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 8 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 8 as defined by this region descriptor.
9 M9EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 9 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 9 as defined by this region descriptor.
10 M10EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 10 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 10 as defined by this region descriptor.
11 M11EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 11 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 11 as defined by this region descriptor.
12 M12EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 12 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 12 as defined by this region descriptor.
13 M13EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM.

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 13 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 13 as defined by this region descriptor.
14 M14EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 14 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 14 as defined by this region descriptor.
15 M15EN	Calibration Remap Descriptor n Master x Enable - These bits determine whether calibration remapping is performed as defined by this descriptor based on the logical master ID of the requesting AHB master. If the current access hits in the remap region defined by this descriptor, and the MxEN field which corresponds to the requesting AHB master is asserted, then calibration remapping is performed and the content is fetched from the overlay RAM. If the current access hits in the remap region defined by this descriptor, but the MxEN field which corresponds to the requesting AHB master is deasserted, then calibration remapping is not performed and the content is fetched from the flash. 0 Calibration remap evaluation is ignored on flash access requests from Master 15 as defined by this region descriptor. 1 Calibration remap evaluation is enabled on flash access requests from Master 15 as defined by this region descriptor.
16–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 CRDSize	Calibration Remap Descriptor *n* Size - This field specifies the size of the calibration remap region. 00000 Reserved 00001 Reserved 00010 Reserved 00011 Reserved 00100 Reserved 00101 Region size is 32 bytes. Low-order 5 bits of LSTARTADDR and PSTARTADDR must be zero 00110 Region size is 64 bytes. Low-order 6 bits of LSTARTADDR and PSTARTADDR must be zero 00111 Region size is 128 bytes. Low-order 7 bits of LSTARTADDR and PSTARTADDR must be zero 01000 Region size is 256 bytes. Low-order 8 bits of LSTARTADDR and PSTARTADDR must be zero 01001 Region size is 512 bytes. Low-order 9 bits of LSTARTADDR and PSTARTADDR must be zero 01010 Region size is 1 KB. Low-order 10 bits of LSTARTADDR and PSTARTADDR must be zero 01011 Region size is 2 KB. Low-order 11 bits of LSTARTADDR and PSTARTADDR must be zero

Table continues on the next page...

PFLASH_PFCRD*n*_Word2 field descriptions (continued)

Field	Description
	01100 Region size is 4 KB. Low-order 12 bits of LSTARTADDR and PSTARTADDR must be zero
	01101 Region size is 8 KB. Low-order 13 bits of LSTARTADDR and PSTARTADDR must be zero
	01110 Region size is 16 KB. Low-order 14 bits of LSTARTADDR and PSTARTADDR must be zero
	01111 Region size is 32 KB. Low-order 15 bits of LSTARTADDR and PSTARTADDR must be zero
	10000 Region size is 64 KB. Low-order 16 bits of LSTARTADDR and PSTARTADDR must be zero
	10001 Region size is 128 KB. Low-order 17 bits of LSTARTADDR and PSTARTADDR must be zero
	10010 Region size is 256 KB. Low-order 18 bits of LSTARTADDR and PSTARTADDR must be zero
	10011 Region size is 512 KB. Low-order 19 bits of LSTARTADDR and PSTARTADDR must be zero
	10100 Region size is 1 MB. Low-order 20 bits of LSTARTADDR and PSTARTADDR must be zero
	10101 Region size is 2 MB. Low-order 21 bits of LSTARTADDR and PSTARTADDR must be zero
	10110 Region size is 4 MB. Low-order 22 bits of LSTARTADDR and PSTARTADDR must be zero
	10111 Region size is 8 MB. Low-order 23 bits of LSTARTADDR and PSTARTADDR must be zero
	11000 Reserved
	11001 Reserved
	11010 Reserved
	11011 Reserved
	11100 Reserved
	11101 Reserved
	11110 Reserved
	11111 Reserved

71.6 Functional description

The flash memory controller interfaces between:

- The AHB system bus ports
- Flash memory array
- System RAM

For accesses targeting the flash array, the flash memory controller generates read and write enables, block selects, array address, write size and write data as inputs to the flash array. The flash memory controller captures read data from the flash array and drives it onto the AHB system bus. Up to four pages of data (256-bit page size) may be buffered in each of two ways of the flash memory controller mini-cache. Lines may be prefetched in advance of being requested, allowing single-cycle (zero AHB wait-states) read data responses on buffer hits.

Several prefetch control algorithms are available for controlling line read buffer fills. Prefetch triggering may be restricted to instruction accesses only, data accesses only, or may be unrestricted. Prefetch triggering may also be controlled on a per-master basis.

Buffers may also be selectively enabled or disabled for allocation by instruction and data prefetch.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

71.6.1 Basic interface protocol

Read accesses are terminated under control of the appropriate wait state settings. Thus, the access time of the operation is determined by the setting of the PFCR1[RWSC] field. Access timing can be varied to account for the operating conditions of the SoC (frequency, voltage, temp) by appropriately setting the PFCR1[RWSC] field.

71.6.2 Access protections

On-chip Flash memory accesses are subject to restrictions based on chip security implementation and/or requirements based on functional restrictions of the flash memory, e.g., a read access to a flash block while it is currently being written is not allowed. When a prohibited access is attempted, a system bus error termination results.

The flash memory controller may invoke a system bus error termination in the following scenarios:

- Attempted access by an AHB master whose corresponding read access control or write access control settings do not allow the access, thus causing a protection violation. See Platform Flash Access Protection Register (PFLASH_PFAPR) for more detail. In this case the flash memory controller does not initiate the requested flash array access.
- Attempted instruction fetch by an AHB master to the BAF flash region when the PFCR3[BAF_DISABLE] field in Platform Flash Configuration Register 3 (PFCR3) is set. The assertion of PFCR3[BAF_DISABLE] indicates that code stored in the BAF region should not be executed. See Platform Flash Configuration Register 3 (PFLASH_PFCR3) for more detail. In this case, the flash memory controller does not initiate a flash array access.
- Attempted access by an AHB master to a flash region where the corresponding censorship control is asserted.
- The flash returns an error response on an attempted access to a partition that is unavailable. (see Flash error response operation or Embedded Flash Memory).
- Attempted access by an AHB master to a reserved region in the flash memory map.

The flash array may also signal an error response to terminate a requested access due to improper sequencing during program/erase operations, improper sequencing during array integrity testing. When an error response is received the flash memory controller does not

update or validate a page read buffer. An error response may be signaled on a read or interlock write operation. For more information on the specifics related to signaling of flash errors, including flash ECC events, array integrity testing and read-while-write events, refer to the flash memory chapter.

71.6.2.1 Censorship

The flash space is defined by regions:

- Code flash (instruction and constant data)
- Data flash (data for EEPROM emulation)
- Secure code flash (instruction for security module)
- Secure data flash (data for security module)
- UTEST flash
- Security flash blocks

Each section can be independently censored, and the flash memory controller provides independent censorship control inputs for each region on a read vs. write basis.

71.6.2.2 PFAPR - Platform Flash Access Protection Register (PFAPR)

The flash memory controller provides programmable, configurable access protections for both read and write cycles on a per-master basis via the PFlash Access Protection Register (PFAPR). It allows restriction of read and write requests on a per-master basis. This functionality is described in Platform Flash Access Protection Register (PFLASH_PFAPR). Detection of a protection violation results in an error response from the flash memory controller on the AHB transfer.

71.6.2.3 BAF execution disable

The flash memory controller provides a mechanism to disable instruction execution of the BAF (Boot Assist Flash) code. Setting the BAF Disable field (BAF_DIS) in the PFlash Configuration Register 3 (PFCR3) prevents execution of the BAF instructions. When this field is set, all incoming requests from the system bus on Port0, Port1, or Port2 are evaluated. If the flash memory controller detects an attempted instruction fetch access to the BAF region of the flash, the flash memory controller aborts the access and returns a system bus error. Data accesses to the BAF region are not affected by the state of PFCR3[BAF_DIS].

71.6.3 Security module exclusive control

During "Production at OEM" lifecycle state and beyond, the HSM can establish the secure flash blocks to be exclusive to the HSM and route program or erasing of secure flash blocks to the alternate programming interface.

The secure flash space is independently controlled for each of the following regions:

- 64 KB secure code flash blockA at address range 0x0061_0000 - 0x0061_FFFF
- 64 KB secure code flash blockB at address range 0x0062_0000 - 0x0062_FFFF
- Two 16 KB secure data flash blocks at address range 0x00F8_0000 - 0x00F8_7FFF

The HSM provides flags to individually control the HSM-exclusivity for each of the secure flash regions. When a given secure flash block is marked as exclusive to the HSM, the HSM is the only system bus master granted access to that block and access attempts by non-HSM masters are terminated with error. Once HSM-exclusivity is established for a given secure flash block, the HSM is guaranteed access to that block, regardless of the state of the censorship controls. Secure flash exclusivity applies to read and write accesses.

71.6.4 Security Module Alternate Programming Interface

The secure flash space is independently controlled for each of the following regions:

- 64 KB secure code flash block2 at address range 0x0061_0000 - 0x0061_FFFF
- 64 KB secure code flash block3 at address range 0x0062_0000 - 0x0062_FFFF
- Two 16 KB secure data flash blocks at address range 0x00F8_0000 - 0x00F8_7FFF

The HSM provides flags to individually control the routing interlock write for each of the four secure flash regions to either the main program/erase interface or the alternate program/erase interface. When a given secure flash block is assigned to the alternate program interface, the flash controller appropriately routes incoming interlock write requests within the given secure flash block address range to the flash's alternate program interface. Alternate program control can be controlled independent of HSM exclusivity for a given secure flash block

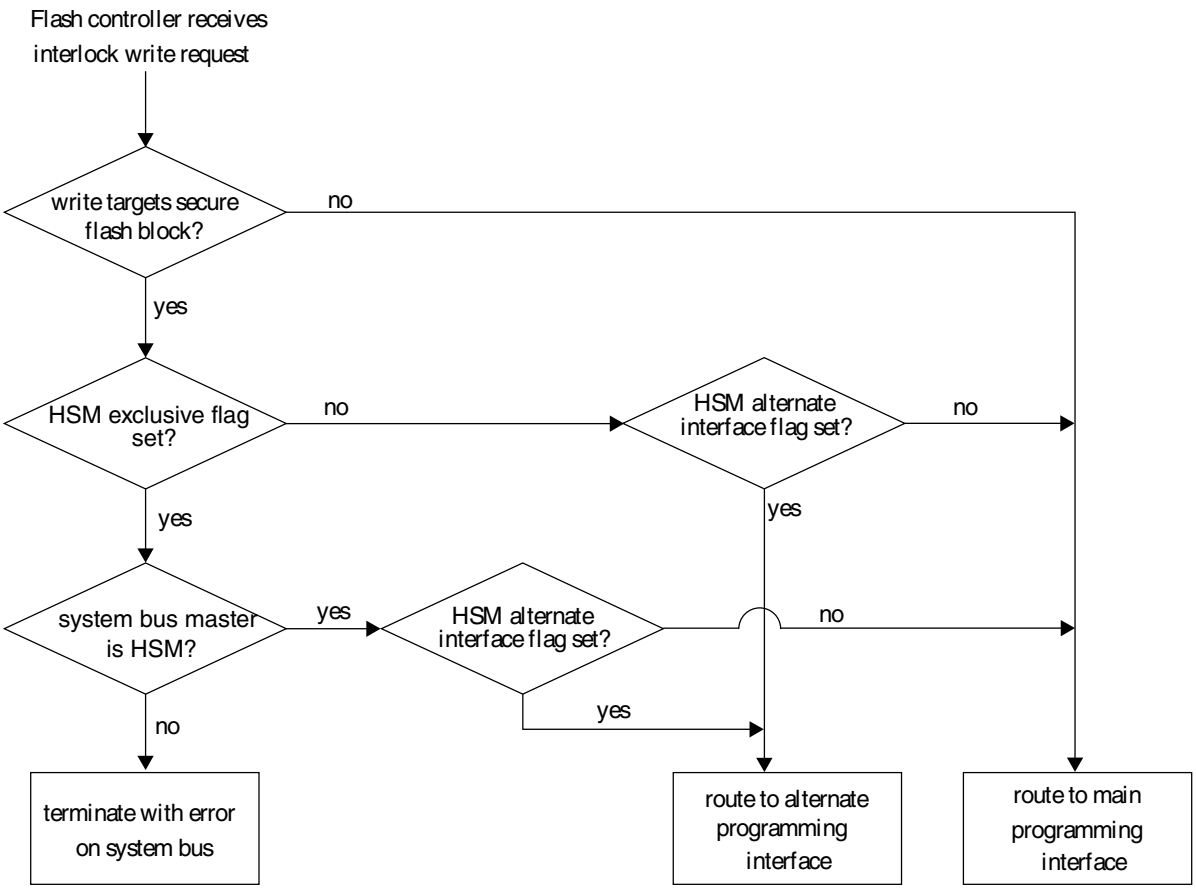


Figure 71-2. Interlock Write Alternate Programming Interface Routing Flowchart

71.6.5 Access pipelining

Accesses to the flash array may be pipelined by driving a subsequent access address and control signals while waiting for the current access to complete. Pipelined access requests are always run to completion and are not aborted by the flash memory controller. Flash access pipelining allows for improved performance by reducing the access latency seen by the AHB master. Access pipelining may be applied only to read cycles targeting the flash array. Address pipelining is enabled by setting the PFCR1[APC] field.

Access pipelining is only supported for normal flash access and is not supported for calibration overlay RAM fetches. On calibration reads, the setting of PFCR1[APC] is ignored.

NOTE

Not all combinations of PFCR1[APC] (Address Pipeline Control) and PFCR1[RWSC] (Read Wait State Control) settings are recommended or supported. Please refer to the device data sheet for guidance.

71.6.6 Line read buffers and prefetch operation

The AHB ports of the flash memory controller each contain a two-way set-associative mini cache, where each way contains four page buffers which are used to hold data read from the flash arrays. Each 256-bit buffer operates independently, and is filled using a single array access. The buffers are used for both prefetch and normal demand fetches.

Prefetch triggering is controllable on a per-master and access-type basis (see Platform Flash Configuration Register 1 (PFLASH_PFCR1)). Bus masters may be enabled or disabled from triggering prefetches, and triggering may be further restricted based on whether a read access is for instruction or data. A read access by the flash memory controller may trigger a prefetch to the next sequential line of array data on the cycle following the request. The access address is incremented by 32-bytes, and a subsequent flash access is initiated. A flash array prefetch is initiated if the data is not already resident in a line read buffer. Prefetched data is always loaded into the least-recently-used buffer.

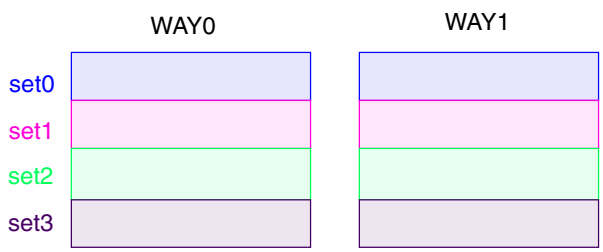


Figure 71-3. Flash memory controller 4-entry, 2-way mini-cache organization

Once the candidate line buffer has been selected, the flash array is accessed and read data loaded into the buffer. If the buffer load was in response to a miss, the buffer which was loaded is immediately marked as most-recently-used. If the buffer load was in response to a speculative fetch to the next-sequential line address after a buffer hit, *the recently-used status is not changed*. Rather, it is marked as most-recently-used only after a subsequent buffer hit.

This policy maximizes performance based on reference patterns of flash accesses and allows for prefetched data to remain valid when non-prefetch enabled bus masters are granted flash access.

Several algorithms are available for prefetch control which trade off performance for power. They are described in Platform Flash Configuration Register 1 (PFLASH_PFCR1). More aggressive prefetching may increase power due to the number of potentially discarded prefetches, but may increase performance by lowering average read latency.

For prefetching to occur, all of the following must apply:

1. PFCR{1,2,4}[P{0,1,2}_BFEN] must be set to '1',
2. PFCR{1,2,4}[P{0,1,2}_PFLIM] must be non-zero, and
3. Either PFCR{1,2,4}[P{0,1,2}_IPFEN] or PFCR{1,2,4}[P{0,1,2}_DPFEN] must be asserted.

Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls.

71.6.7 Instruction/Data prefetch triggering

Port0 prefetch triggering may be enabled for instruction reads via the PFCR1[IPFEN] control field, while Port0 prefetching for data reads is enabled via the PFCR1[DPFEN] control field. Additionally, the PFCR1[PFLIM] must also be set to enable prefetching on Port0. Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls. Prefetches are never triggered by write cycles.

Port1 prefetch triggering may be enabled for instruction reads via the PFCR2[IPFEN] control field, while Port1 prefetching for data reads is enabled via the PFCR2[DPFEN] control field. Additionally, the PFCR2[PFLIM] must also be set to enable prefetching on Port1. Refer to Platform Flash Configuration Register 2 (PFLASH_PFCR2) for a description of these controls. Prefetches are never triggered by write cycles.

Port2 prefetch triggering may be enabled for instruction reads via the PFCR4[IPFEN] control field, while Port2 prefetching for data reads is enabled via the PFCR4[DPFEN] control field. Additionally, the PFCR4[PFLIM] must also be set to enable prefetching on Port2. Refer to the Platform Flash Configuration Register 4 (PFCR4) section for a description of these controls. Prefetches are never triggered by write cycles.

71.6.8 Per-Master prefetch triggering

Prefetch triggering may be controlled for individual bus masters. Refer to Platform Flash Configuration Register 1 (PFLASH_PFCR1) for a description of these controls.

71.6.9 Buffer allocation

Allocation of the line read buffers is controlled via the PFCR3 control register, specifically the line buffer configuration ({P0,P1,P2}_WCFG) field. Refer to Platform Flash Configuration Register 3 (PFLASH_PFCR3). The buffers can be organized as a

"pool" of available resources (with both ways within a given set) or with a fixed partition between ways allocated to instruction or data accesses. For the fixed partitions, way 0 is allocated for instruction fetches and way 1 for data accesses.

71.6.10 PFlash calibration remap support

The flash memory controller supports calibration development by providing a remapping function to route flash accesses to on-chip System RAM that can be used as overlay RAM.

Additionally, support is provided to reroute flash accesses to other on-chip flash memory addresses.

The overlay function supports the following features:

- Can be mapped over internal flash memory
  - Allows calibration of constant data without requirement for additional external RAMs and calibration memory interfaces
- Protected with error detection and correction mechanism
- Accesses to all overlay RAMs can achieve the same timing as accesses to internal flash
  - Timing for calibration accesses to overlay RAM is only guaranteed if RWSC is set sufficiently high to cover the intrinsic burst access incurred when fetching a full page of calibration data
- Support for up to 32 distinct calibration remap regions.
- Protection against accidental enable and reconfiguration of calibration remap function :
  - Two stage mechanism to enable remap, using both global and individual calibration region descriptor enable bits
  - Optional configuration of calibration region descriptors as redundant pairs, allowing detection of accidental region reconfiguration.
- Support for overlay remapping to unique calibration regions on a per-master basis.

The calibration remap function supports the following overlay targets:

- A portion of the system RAM can be used for overlay.
- Addresses within any non-dedicated block of on-chip flash memory. Secure flash, UTEST flash, and BAF cannot be used as overlay targets.

71.6.10.1 PFlash calibration remap to RAM

The overlay RAM is selected based on the translated physical address.

The translated physical address defines the destination memory. Table 71-1 shows the base addresses for the calibration data memories as defined by the system memory map.

Table 71-1. Calibration data memory base addresses

Calibration Memory	Base Address	Possible Access Sources
Flash Arrays	0x0{8,9,A}--_----	P0, P1
System RAM	0x40{0,1}-_----	P0, P1

The system RAM connection from the calibration remap logic is implemented via a private 64-bit connection to the platform RAM controller.

Accesses to the flash arrays always use the logical, non-remapped system address. Calibration data store references to the system RAM always use the physical, non-remapped system address. Conversely, calibration data load references can be accessed using a translated physical address or, if provided by the requesting master, the physical, non-remapped system address.

The read data interface of system RAM is 64 bits wide , whereas the flash read data interface is 256 bits wide. As a result, a remapped calibration load access initiates a four-beat burst sequence to mimic the throughput of a single flash access. There is a direct relationship between the operating frequency range and the required number of wait states to correctly sample flash read data. By contrast, a remapped calibration load access has a fixed, intrinsic 4-beat latency. Therefore, in order for an initial, random calibration load access to mimic the flash access time, PFCR1[RWSC] read wait state setting must be programmed to a value which sufficiently covers the intrinsic access time to complete a four-beat burst sequence from the slowest available overlay target at the specified device operating frequency. Expressed mathematically:

$$(\mathrm{RWSC}+2) \geq \left(\frac{\mathrm{IntrinsicFlashAccessTime}}{\mathrm{OperatingFrequencyClockPeriod}}+2\right) \geq (\mathrm{OverlayRAMTransferBurstLatency})$$

Equation 45. Access time

Notice the overall latency on a random, initial flash access that does not hit in the prefetch mini-buffer incurs an additional cycle latency beyond the intrinsic flash access time. This cycle accounts for pipeline stalls incurred by the flash controller state machine. The formula above is based on specified intrinsic flash access times of the C55FMC array at 150 °C. Calibration data that is loaded in the flash controller's mini-cache is returned with single-cycle latency.

The memory-mapped calibration remap (region) descriptor (CRD*n*) registers define the required address translation. Each calibration remap descriptor includes the logical base address, the translated physical base address and a region size (plus other control bits). The calibration remap evaluation is initiated when the flash controller is presented with a

flash access request from any system bus master to a location in the mirrored flash address space. Conversely, references using the associated non-mirrored flash system address are not evaluated for calibration remapping.

The address translation mechanism requires the logical and physical base addresses of the region be aligned on 0-modulo-size boundaries.

The region size is decoded to create the appropriate address bit enables which are then applied to the registered logical flash address. An equality comparator then compares the adjusted logical flash address against the logical address from the calibration remap descriptor to determine the region "hit". The hit determination is further qualified based on the bus master number that initiated the flash access. At the same time, the logical flash address and the physical address defined in the calibration remap descriptor are merged to form the translated address to be used in the event of a calibration region hit. Aside from the individual instantiations of the calibration remap descriptors, this logic examines all the descriptor hit indicators to select the appropriate translated address.

In the absence of a region "hit" data is returned from the flash at the provided logical address.

It should be noted in the event of *overlapping remap regions*, the calibration remap descriptor with the largest number is used for the address translation, that is, if a logical flash address hits in multiple descriptors, say CRDx and CRDy, the translated address from CRDy is used, given y > x.

In general, the calibration remap logic operates only on flash *data* accesses. However, flash instruction references can optionally remapped while in debug modes of operation to support the use of software breakpoints when PFCRC[IRMEN] is set.

71.6.10.2 PFlash calibration remap to flash memory

Calibration remapping of a flash access to another on-chip flash address uses the same mechanism and controls as remapping to RAM addresses. The following should be noted:

- UTEST flash, BAF, and secure flash blocks cannot be remapped nor can accesses to other flash blocks be rerouted to addresses in UTEST flash, BAF, or secure flash
- All other flash blocks can be overlaid using the *mirrored* address range[1].

1. A *mirror* address for an address within a flash block is calculated by setting bit 27 (where the least significant bit of the address field is numbered 0).

71.6.11 Reliability considerations

This section summarizes mechanism to enhance the reliability of flash by correcting and detecting faults.

71.6.11.1 e2eECC End-to-End ECC

71.6.11.1.1 e2eECC and flash accesses

There are multiple complications associated with the use of the standard e2eECC algorithm with flash memory. The basic issues involve: the default state of a flash block that is erased (all ones) and the fact that programming an erased flash involves changing the state of memory bit from a logical 1 to a logical 0. These factors require that the system level e2eECC must be modified on references to the flash memory in three ways.

First, the all-ones codeword, since it reflects the state of an erased flash location, is treated as a valid error free encoding; in particular, it is required that the checkbits associated with the all all-ones value and the all zeroes data value are required to be the same with a checkbit value of 0xFF. Second, since an erased flash block generates an all-ones data for all locations, the address field cannot be included in the ECC code used by the flash. The flash array implements special address re-encoding logic based on the decoded array address to protect against address faults. Third, there are additional ECC implications associated with the flash memory and its support for EEPROM emulation. Lastly, there are ECC implications associated with the storage and retrieval of overlaid calibration data.

In summary, the basic operation of an NVM bit cell impacts the e2eECC algorithm used in the flash memory in multiple ways. The required ECC adjustments are handled by the platform flash controller before data is written to the flash array(s) or applied to read data accessed from the array(s). These adjustments are two-fold. Consider a flash write during a programming event:

1. Remove the address field from the ECC codeword by XOR'ing the address calculation of the H-matrix from the checkbits.
2. Invert the resulting 8-bit ECC checkbit vector. This step is required to support the all-ones erased state.

On a flash read operation, a similar but "reversed" set of steps are performed as the data is driven into the system bus interconnect:

1. Invert the 8-bit ECC checkbit vector read from the flash.
2. Factor the address field into the ECC checkbits by XOR'ing the address calculation of the H-matrix.

Consider the following flash ECC example. Let the flash block containing address 0x00345678 be erased. The following is the sequence of operations:

1. Read address 0x00345678. Flash array returns fl_rdata = 0xFFFFFFFF_FFFFFFFF, fl_cdata = 0xFF. The flash memory controller calculates the addr_chkbit = 0xC4 using the H-matrix; it inverts the fl_cdata vector to 0x00 and factors in the addr_chkbit = 0xC4 to produce the following valid e2eECC codeword:

   addr = 0x00345678, rdata = 0xFFFFFFFF_FFFFFFFF, rchkbit = 0xC4 (0x00 ^ 0xC4)

2. Program address 0x00345678 with data = 0xBABEFACE_DEADBEEF. The initiating bus master calculates the e2eECC codeword as:

   addr = 0x00345678, wdata = 0xBABEFACE_DEADBEEF, wchkbit = 0xE3

   Before performing the interlock write to the flash array, the flash memory controller adjusts the chkbit value as fl_wchkbit = 0xD8 (0xE3 ^ 0xff ^ 0xC4) by toggling the original write checkbits and then factoring in the addr_ecc. Accordingly, the codeword stored in the flash array is:

   fl_data = 0xBABEFACE_DEADBEEF, fl_checkbit = 0xD8

3. Read address 0x00345678. The flash array returns the stored codeword, fl_rdata = 0xBABEFACE_DEADBEEF, fl_rchkbit = 0xD8. The flash memory controller inverts the checkbit vector and factors in the address checkbits (0xD8 ^ 0xFF ^ 0xC4) to produce:

   addr = 0x00345678, rdata = 0xBABEFACE_DEADBEEF, rchkbit = 0xE3

   This read codeword matches the original e2eECC codeword generated by the write.

Malfunction of the ECC logic described above may result in the corruption of the SEC/DED event reporting. The flash memory controller performs EDC after ECC check on all flash transactions. If a mismatch is detected, indicating a failure in the ECC logic, the event is reported to the device fault collection module.

71.6.11.2 Flash address generation check

Fault detection coverage of the address and data are handled by ECC performed within the flash and e2eECC performed at the master. Fault detection coverage of the address path and control within the flash memory controller rely on a feedback path between the flash controller and flash. Recall on a requested access to flash, the flash memory controller must decode the system AHB bus signals to generate the corresponding flash

interface signals to invoke a flash lookup. In addition to providing the requested read data, the flash also provides output sidebands reflecting the encoded address and block selects used to perform the actual row lookup.

This sideband information is used by the flash memory controller to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the flash memory controller or the transmission path between the flash memory controller and flash array, the event is forwarded to the device fault collection module and the corresponding buffer is invalidated.

71.6.11.2.1 Overlay RAM feedback check

The integrity of the address and data on an overlay RAM transaction is covered by e2eECC check performed by the requesting master. Fault detection coverage of the address path and control within the flash memory controller is handled by a transaction monitor which verifies the integrity of the transactions between the flash memory controller and the on-chip overlay RAM. The function of the transaction monitor relies on a feedback path between the flash memory controller and the on-chip overlay RAM, wherein the RAM provides latched address and control feedback outputs as an indication of received inputs when a RAM access is initiated. This feedback information is used by the flash memory controller transaction monitor to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within the flash memory controller or the transmission path between the flash memory controller and on-chip overlay RAM array, the event is forwarded to the Fault Collection and Control Unit (FCCU).

71.6.11.2.2 Reliability considerations on overlay accesses

Because calibration write accesses are always presented with the physical, non-remapped system address, the write data checkbits presented by the master are calculated based on the physical overlay RAM system address. On a subsequent overlay read, the ECC checkbits must be manipulated to replace the physical overlay RAM address contribution with the logical flash address contribution. This is required to satisfy the e2eECC check at the originating master.

In addition, the flash memory controller can be configured to treat calibration RAM space as a safety-critical component. By programming PFCRCR[SAFE_CAL]=1, the flash memory controller treats the set of available calibration region descriptors as a replicated pair, with each set containing half the total number of available calibration region descriptors.

If there are 32 calibration region descriptors, CRD0 – CRD31, and PFCRCR[SAFE_CAL]=1, the flash memory controller treats CRD0 – CRD15 and CRD16 – 32 as replicated sets of 16 calibration region descriptors, where:

- CRD0 and CRD16 are configured identically
- CRD1 and CRD17 are configured identically
- CRD2 and CRD18 are configured identically
- ...
- and CRD15 and CRD31 are configured identically

Expressed more generally, the contents of CRD[*n*] must be identical to the contents of CRD[(N/2) + *n*], where N is the total number of region descriptors and $0 \le n \le (N/2) - 1$.

Note

It is the user's responsibility to establish redundant calibration region descriptors by programming the associated pairs of CRDs.

Incoming flash read requests are compared simultaneously against the calibration regions defined in CRD0 to CRD[(N/2) – 1] and the calibration regions defined in CRD[N/2] to CRD[N – 1]. Due to the enabled redundancy, the parallel overlay remap hardware structures should evaluate to symmetric results. If a mismatch is detected, the event is reported to the FCCU and calibration remapping is not performed. Instead, data is returned from the flash.

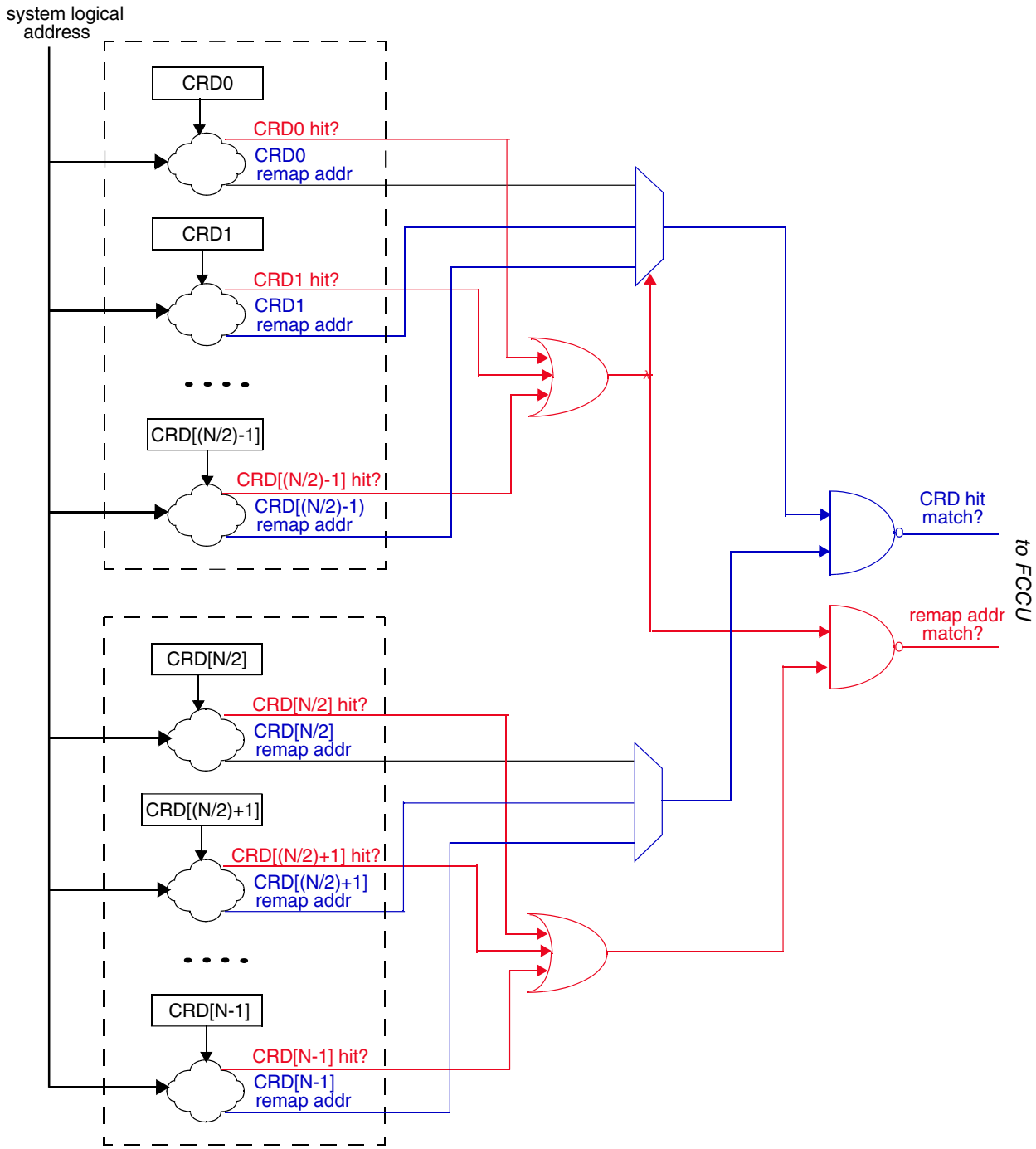


Figure 71-4. Safety-critical calibration remap datapath with redundancy

71.6.12 ECC on data flash accesses

For single-bit correction events, the corrected data and checkbits are returned to the requesting master, and the single-bit correction event is reported to the MEMU.

In the event of a non-correctable error detection, a fixed, illegal opcode value (1555_1555h) is returned to the requesting master along with the associated ECC checkbits as determined by the requesting address and the non-correctable error event is reported to the MEMU.

NOTE

EEPROM should be avoided for storage of executable code.

71.6.13 Array integrity considerations

During an array integrity sequence, the flash memory array ignores any incoming read requests. When a flash array integrity check is in progress, the flash memory controller terminates all flash access requests with an error. More specifically, it aborts the incoming flash access requests and terminates the system bus transfer with an error.

Chapter 72
Embedded Flash Memory

72.1 Chip-specific Embedded Flash memory information

72.1.1 Embedded Flash Memory configuration

This section describes the configuration of this module specific to the MPC5746C.

72.1.2 Register bit mapping

The C55FMC_LOCKn registers have the same bit-to-block mapping as the C55FMC_SELn registers.

72.1.3 C55FMC_LOCK0 and C55FMC_SEL0 register bit mapping

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	TSLOCK	ATSL	LOWLOCK[13:0]														MIDLOCK[15:0]															
End address	0x00403FFF						0x0062FFFF	0x00FFFFFF	0x0061FFFF	0x00FEFFFF	0x00FD7FFF	0x00FD7FFF	0x00FC7FFF	0x00FC7FFF	0x00407FFF	0x00F8FFFF							0x00FBFFFF	0x00FB7FFF	0x00FAFFFF	0x00FABFFF	0x00FA7FFF	0x00FA3FFF	0x00F9FFFF	0x00F9BFFF	0x00F97FFF	0x00F93FFF
Start address	0x00400000						0x00620000	0x00FF0000	0x00610000	0x00FE0000	0x00FD8000	0x00FD0000	0x00FC8000	0x00FC0000	0x00404000	0x00F8C000							0x00FB8000	0x00FB0000	0x00FAC000	0x00FA8000	0x00FA4000	0x00FA0000	0x00F9C000	0x00F98000	0x00F94000	0x00F90000

Table continues on the next page...

Flash Block Name	Read Lock Regions
UTest NVM Block Space 16 KB	Read Lock Region 0
	—
	—
	—
	—
	—
64 KB HSMCode Block3 (RWW partition 1)	Read Lock Region 3
64 KB Code Flash Block1 (RWW partition 1)	Read Lock Region 1
64 KB HSMCode Block2 (RWW partition 0)	Read Lock Region 3
64 KB Code Flash Block0 (RWW partition 0)	Read Lock Region 1
32 KB Code Flash Block3 (RWW partition 1)	
32 KB Code Flash Block2 (RWW partition 1)	
32 KB Code Flash Block1 (RWW partition 0)	
32 KB Code Flash Block0 (RWW partition 0)	
16 KB BAF Block (RWW partition 1)	—
16 KB Code Flash Block 0 (RWW partition 0)	Read Lock Region 1
	—
	—
	—
	—
	—
	—
32 KB Code Flash Block1 (RWW partition 3)	Read Lock Region 1
32 KB Code Flash Block 0 (RWW partition 2)	
16 KB Code Flash Block 7 (RWW partition 3)	
16 KB Code Flash Block 6 (RWW partition 3)	
16 KB Code Flash Block5 (RWW partition 3)	
16 KB Code Flash Block4 (RWW partition 3)	
16 KB Code Flash Block3 (RWW partition 2)	
16 KB Code Flash Block 2 (RWW partition 2)	
16 KB Code Flash Block 1 (RWW partition 2)	
16 KB Code Flash Block 0 (RWW partition 2)	

72.1.4 C55FMC_LOCK1 and C55FMC_SEL1 register bit mapping

Start address	End address	Register Field	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
		Reserved																	HIGHLOCK[15:0]																
0x00F84000	0x00F87FFF		—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—		
0x00F80000	0x00F83FFF																																	

Table continues on the next page...

Flash Block Name	Read Lock Regions
16KB HSM Data Block 1(RWW partition 5)	Read Lock Region 4
16KB HSM Data Block 0(RWW partition 4)	

72.1.5 C55FMC_LOCK2 and C55FMC_SEL2 register bit mapping

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	256LCK_L[31:0]																															
End address	—	—	—	—	—	—	—	—	—	—	0x0157FFFF	0x0153FFF	0x014FFFFF	0x014BFFFF	0x0147FFFF	0x0143FFFF	0x013FFFFF	0x013BFFFF	0x0137FFFF	0x0133FFFF	0x012FFFFF	0x012BFFFF	0x0127FFFF	0x0123FFFF	0x011FFFFF	0x011BFFFF	0x0117FFFF	0x0113FFFF	0x010FFFFF	0x010BFFFF	0x0107FFFF	0x0103FFFF
Start address											0x01540000	0x01500000	0x014C0000	0x01480000	0x01440000	0x01400000	0x013C0000	0x01380000	0x01340000	0x01300000	0x012C0000	0x01280000	0x01240000	0x01200000	0x011C0000	0x01180000	0x01140000	0x01100000	0x010C0000	0x01080000	0x01040000	0x01000000

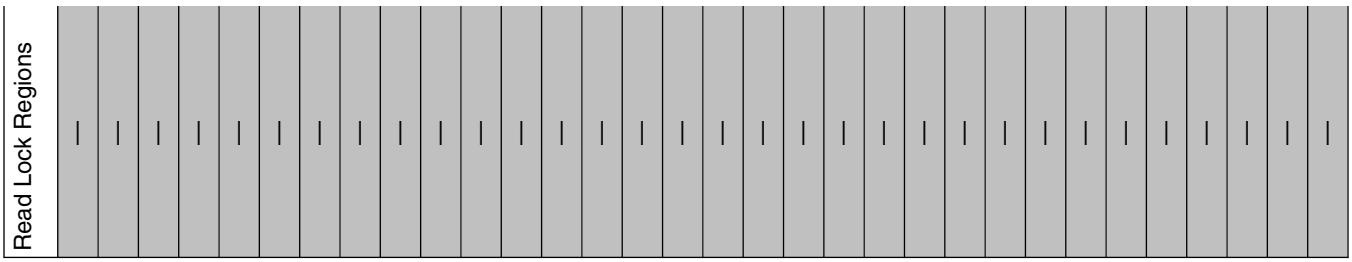
Table continues on the next page...

Read Lock Regions	Flash Block Name
—	256 KB Code Flash block21(RWW partition 9)
—	256 KB Code Flash block20(RWW partition 9)
—	256 KB Code Flash block19(RWW partition 9)
—	256 KB Code Flash block18(RWW partition 8)
—	256 KB Code Flash block17(RWW partition 8)
—	256 KB Code Flash block16(RWW partition 8)
—	256 KB Code Flash block15(RWW partition 7)
—	256 KB Code Flash block14(RWW partition 7)
—	256 KB Code Flash block13(RWW partition 7)
—	256 KB Code Flash block12(RWW partition 7)
—	256 KB Code Flash block11(RWW partition 7)
—	256 KB Code Flash block10(RWW partition 7)
—	256 KB Code Flash block9(RWW partition 7)
—	256 KB Code Flash block8(RWW partition 7)
—	256 KB Code Flash block7(RWW partition 6)
—	256 KB Code Flash block6(RWW partition 6)
—	256 KB Code Flash block5(RWW partition 6)
—	256 KB Code Flash block4(RWW partition 6)
—	256 KB Code Flash block3(RWW partition 6)
—	256 KB Code Flash block2(RWW partition 6)
—	256 KB Code Flash block1(RWW partition 6)
Read Lock Region 1	256 KB Code Flash block0(RWW partition 6)

72.1.6 C55FMC_LOCK3 and C55FMC_SEL3 register bit mapping

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Bit Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	LOCK_U_256K[15:0]															
Field or Flash Block Name	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table continues on the next page...



72.1.7 Embedded Flash Memory register reset value

All register reset values for the FMC registers that are device specific are shown below. All other registers reset values are shown in the Embedded Flash Memory chapter

Module Configuration Register (FMC_MCR) register resets to 0000_0600h.

Extended Module Configuration Register (FMC_MCRE) register resets to 0B01_0C51h.

Over-Program Protection 0 register (FMC_OPP0) register resets to 0000_0000h.

Over-Program Protection 1 register (FMC_OPP1) register resets to 0000_0000h.

Over-Program Protection 2 register (FMC_OPP2) register resets to 0000_0000h.

Over-Program Protection 3 register (FMC_OPP3) register resets to 0000_0000h.

NOTE

Do not access the reserved area of the C55FMC memory map.

72.2 Introduction

The primary function of the embedded flash memory is to serve as electrically programmable and erasable non-volatile memory (NVM) that may be used for instruction or data storage.

The following figure shows the top-level diagram and functional organization of the flash memory unit.

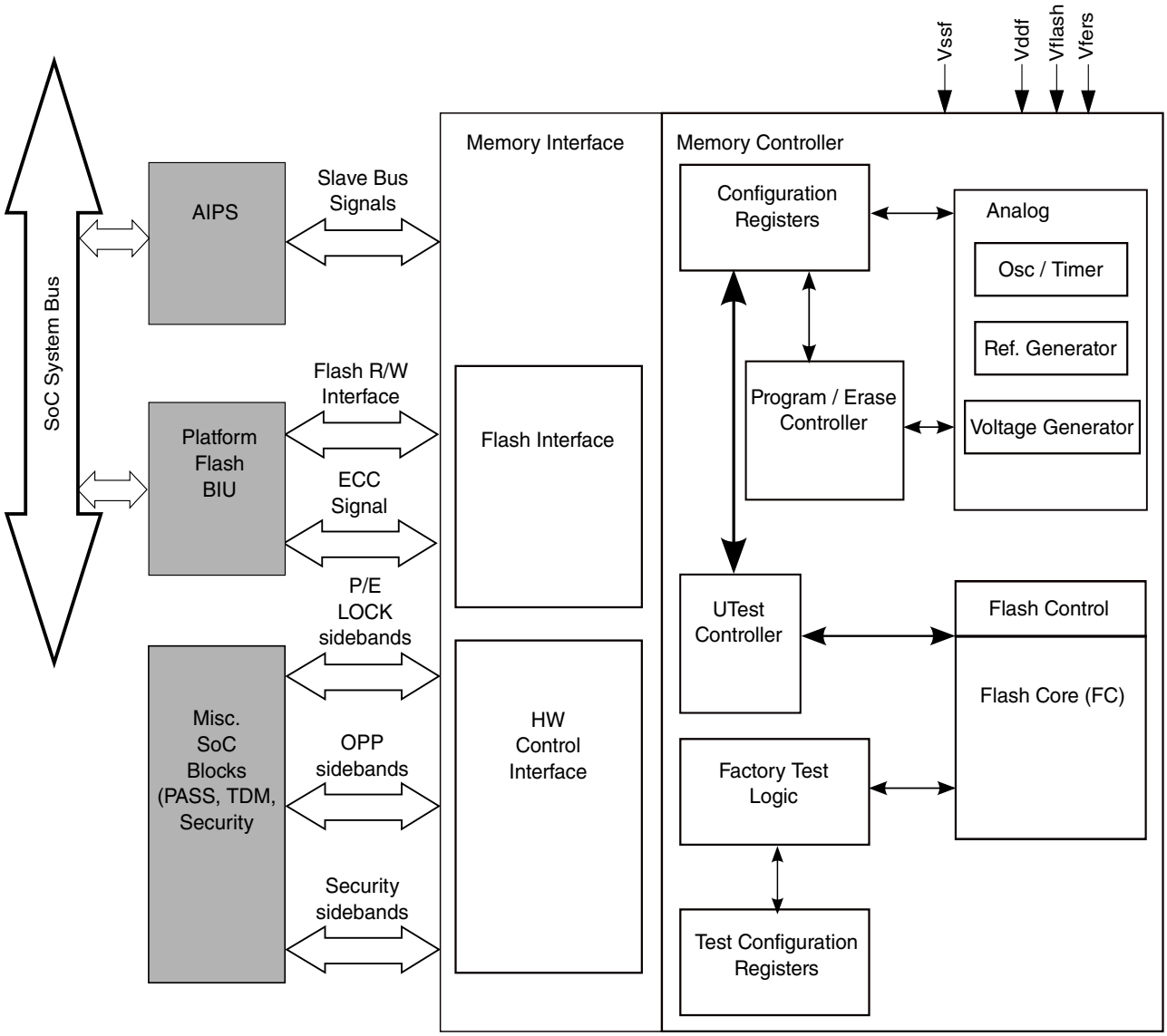


Figure 72-1. c55fmc block diagram

72.2.1 Overview

The embedded flash memory is addressable by word (32 bits) or double word (64 bits) for program operation, and page (256 bits) for read operation. Multiple word or double-word writes may be done to the flash memory to fill up the program page buffer (256 bits), enabling page programming (256 bits, requiring 4 double-word writes) and quad-page programming (1024 bits, requiring 16 double-word writes). Flash memory reads always return 256 bits, although read page buffering may be performed by the Bus Interface Unit (BIU).

For more details on the embedded flash memory architecture and features, see Functional Description.

72.2.2 Features

The embedded flash memory includes these distinct features:

- Test information stored in a dedicated nonvolatile block (referred to as the UTest block)
- OTP space made available in the UTest block
- Read page size of 256 bits (8 words)
- ECC with single-bit correction, double-bit detection (all 1's valid)
- Quad Page programming (64-bit granularity)
- Software programmable block program/erase restriction control
- Erasing of selected block(s)
- Independent programming of the UTest NVM block
- Embedded hardware program and erase algorithms
- Support for reading while writing when the accesses are to different partitions
- Erase suspend, program suspend, and erase-suspended program
- UTest mode (user-accessible test modes) including Array Integrity and Margin Read
- Triple Voted Flops for flash functions requiring high reliability, e.g., internal trimming, redundancy, and mode control

72.2.3 Modes of operation

Following is a brief description of the embedded flash memory operating modes.

- *User mode* is the default operating mode of the embedded flash memory. In this mode, it is possible to read and write registers, read and interlock write the memory array, program the memory array, and erase the memory array. In this mode program and erase operations are initiated by doing array and register writes, and are controlled by an internal state machine.
- *Low power mode* turns off all DC current sources within the embedded flash memory on the Vflash domain, and enables VDDF to be power gated. The embedded flash memory is not accessible for read, write, program, or erase when in low power mode.
- *UTest mode* is a tiered test mode strategy in which a portion of the factory test modes are made available. This mode is protected but accessible.

72.3 UTest NVM block

The UTest NVM block may be enabled by the BIU. When the UTest NVM space is enabled, all operations are mapped to the UTest NVM block. User-mode programming of the UTest NVM block is enabled only when MCR[PEAS] is high. The UTest NVM block is an OTP block (assuming Test Mode Disable Seal is written) - thus erase is not allowed.

The UTest NVM block supports RWW, and is grouped with the blocks in partition 0.

The UTest NVM block may be locked against program by using the lock registers.

Programming of the UTest NVM space has restrictions that are similar to those of the main space in terms of how ECC is calculated. Only one program operation is allowed per 64-bit ECC segment.

The UTest NVM block contains specified data that is needed for embedded flash memory or SoC features.

72.4 Test mode disable seal

Also included in the UTest NVM block is a mechanism to disable factory entry into Test mode. Extreme care must be taken when using this feature, as blocks that are selected to be protected in this method will not be able to have possible failures analyzed by factory failure analysts. Once sealed, the UTest block becomes OTP (Erase Locked, and Over Program Protection enabled). The UTest Block also is not accessible in Test mode.

Protection of this sort prevents all high-voltage operations to the flash executed by the internal state-machine, as well as reads through this state machine, and reads through the Array Integrity state machine when using Test mode interfaces.

UTest operations Margin Read and Array Integrity are also protected by preventing MISR updates on blocks selected for protection, although reads will still be done and single bit corrections and double bit detections will be logged. Protection through other interfaces is protected by normal User mode protection mechanisms, and are device-specific.

The method to disable factory entry into Test mode is to first program the Test Mode Disable Seal location to be 0x2D3C_4B5A. Once the next reset is asserted, Test mode will be disabled.

It is possible to create a password to enable factory entry into test mode. This can be programmed into the Test Mode Disable Override Passcode. The passcode may not be 0x0000_0000, 0xFFFF_FFFF, or 0x5555_5555. These are all invalid passcodes and will not be accepted to override. If it is desired by the customer that override never be possible, one of the three invalid passcodes should be put into this location. Passcode may be entered to authenticate entry (if enabled) by doing a 32-bit register write to register address 0x90. The UTest NVM block is always protected even if the Test Mode Disable Seal password is written. UTest operations Margin Read and Array Integrity are always protected even if the Test Mode Disable Seal password is written.

Only blocks selected in the Test Mode Disable Block Select field(s) are controlled by the Test Mode Disable feature. Thus it is possible for customers to selectively pick blocks that have this type of protection, and will not be eligible for factory failure analysis. Bits programmed to 0 in the Test Mode Disable Block Select field(s) will designate blocks that are controlled by the Test Mode Disable feature. If either Test Mode Disable Block Select - 0 or Test Mode Disable Block Select - 1 have a 0 programmed, that block will be protected. In order to provide two opportunities to select blocks for the Test Mode Disable, two regions are available, Test Mode Disable Block Select - 0 and Test Mode Disable Block Select - 1. The bit of these two regions are logically ANDed, to define which blocks are effectively selected for the Test Mode Disable feature. The Block Select field is organized as follows, and aligns with how blocks are defined in the LOCK registers.

Table 72-1. Test mode disable block select

Block	Bits used to select blocks
Low Block Disable	Data[13:0]
Unused	Data[15:14]
Mid Block Disable	Data[31:16]
High Block Disable	Data[47:32]
256K Block Disable	Data[95:48]

72.5 Memory map and register definition

The embedded flash memory map consists of a flash memory array (which includes main array space and UTest NVM space) and a region of registers associated with the programming model that enable flash memory array operation and modification.

The address space consists of 16 KB, 32 KB, 64 KB, and 256 KB blocks.

C55FMC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Module Configuration Register (C55FMC_MCR)	32	R/W	0000_0600h	72.5.1/3045
8	Extended Module Configuration Register (C55FMC_MCRE)	32	R	See section	72.5.2/3050
10	Lock 0 register (C55FMC_LOCK0)	32	R/W	BFFF_FFFFh	72.5.3/3053
14	Lock 1 register (C55FMC_LOCK1)	32	R/W	0000_FFFFh	72.5.4/3055
18	Lock 2 register (C55FMC_LOCK2)	32	R/W	FFFF_FFFFh	72.5.5/3055
1C	Lock 3 register (C55FMC_LOCK3)	32	R/W	0000_FFFFh	72.5.6/3056
38	Select 0 register (C55FMC_SEL0)	32	R/W	0000_0000h	72.5.7/3057
3C	Select 1 register (C55FMC_SEL1)	32	R/W	0000_0000h	72.5.8/3058
40	Select 2 register (C55FMC_SEL2)	32	R/W	0000_0000h	72.5.9/3059
44	Select 3 register (C55FMC_SEL3)	32	R/W	0000_0000h	72.5.10/ 3060
50	Address register (C55FMC_ADR)	32	R	0000_0000h	72.5.11/ 3061
54	UTest 0 register (C55FMC_UT0)	32	R/W	0000_0001h	72.5.12/ 3063
58	UMISR register (C55FMC_UM0)	32	R/W	0000_0000h	72.5.13/ 3066
5C	UMISR register (C55FMC_UM1)	32	R/W	0000_0000h	72.5.13/ 3066
60	UMISR register (C55FMC_UM2)	32	R/W	0000_0000h	72.5.13/ 3066
64	UMISR register (C55FMC_UM3)	32	R/W	0000_0000h	72.5.13/ 3066
68	UMISR register (C55FMC_UM4)	32	R/W	0000_0000h	72.5.13/ 3066
6C	UMISR register (C55FMC_UM5)	32	R/W	0000_0000h	72.5.13/ 3066
70	UMISR register (C55FMC_UM6)	32	R/W	0000_0000h	72.5.13/ 3066
74	UMISR register (C55FMC_UM7)	32	R/W	0000_0000h	72.5.13/ 3066
78	UMISR register (C55FMC_UM8)	32	R/W	0000_0000h	72.5.13/ 3066
7C	UMISR register (C55FMC_UM9)	32	R/W	0000_0000h	72.5.14/ 3067
80	Over-Program Protection 0 register (C55FMC_OPP0)	32	R	See section	72.5.15/ 3068
84	Over-Program Protection 1 register (C55FMC_OPP1)	32	R	See section	72.5.16/ 3069
88	Over-Program Protection 2 register (C55FMC_OPP2)	32	R	See section	72.5.17/ 3070
8C	Over-Program Protection 3 register (C55FMC_OPP3)	32	R	See section	72.5.18/ 3070
90	Test Mode Disable Password Check register (C55FMC_TMD)	32	R/W	0000_0000h	72.5.19/ 3071

72.5.1 Module Configuration Register (C55FMC_MCR)

NOTE

1. A number of Module Configuration Register (MCR) bits are locked against write by other bits. These locks are discussed in relationship to each bit in this section. Simultaneously writing bits which lock each other out is also discussed in Functional Description.
2. See Functional Description for information about simultaneous MCR writes, and priority levels.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RVE	RRE	AEE	EEE	0											
W	w1c	w1c	w1c	w1c												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EER	RWE	SBC	0	PEAS	DONE	PEG	PECIE	0			PGM	PSUS	ERS	ESUS	EHV
W	w1c	w1c	w1c													
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

C55FMC_MCR field descriptions

Field	Description
0 RVE	Read Voltage Error RVE provides information on previous reads monitoring the read pump voltage. If the read voltage is detected to be out of range, this bit is set to indicate that previous reads requested may have been

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without voltage issues 1 A previous read may have been corrupted due to read voltage being out of range
1 RRE	Read Reference Error RRE provides information on previous reads monitoring the read reference. If the read reference is detected to be out of range, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without reference issues 1 A previous read may have been corrupted due to read reference being out of range
2 AEE	Address Encode Error AEE provides information on previous reads monitoring the address encode feature. On every read request to the flash, the incoming address is compared to an encoded address (row, column, and block) coming back from the memory array using the read data sense amplifier timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error will be recorded. If an address encode mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without address encode mismatches 1 A previous read may be corrupted based on address encode mismatch
3 EEE	ECC after ECC Error EEE provides information on previous reads monitoring the ECC after ECC feature. On every read request to the flash, ECC is recalculated serially, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error will be reported. If an ECC after ECC mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without ECC after ECC mismatches 1 A previous read may be corrupted based on ECC calculation errors
4–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 EER	ECC Event Error This bit provides information on previous reads. If a double bit detection occurred, the EER bit is set to a '1'. This bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be set by software. In the event of a single bit detection and correction, this bit is not set. If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally. 1 An ECC Error occurred during a previous read.
17 RWE	Read-While-Write Event Error This bit provides information on previous read-while-write (RWW) reads. If an RWW error occurs, this bit is set to 1. The bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be written to a 1. If RWE is not set, or remains 0, this indicates that all previous RWW reads (from the last

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	reset, or clearing of RWE) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring normally. 1 A read-while-write error occurred during a previous read.
18 SBC	Single Bit Correction SBC provides information on previous reads, if the SBCE is set. If a single bit correction occurred, the SBC bit is set to a 1. This bit must then be cleared, or a reset must occur before this bit returns to a 0 state. If SBC is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of SBC) did not require a correction. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect. 0 Reads are occurring without corrections. 1 A Single Bit Correction occurred during a previous read.
19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 PEAS	Program Access Space PEAS indicates which space is valid for program or erase operations — either main array space or UTest NVM space. It should be noted that if the flash is sealed, and the Test Mode Disable Seal is programmed, erase is not allowed on the UTest NVM space. PEAS = 0 indicates the main address space is active for all program or erase operations. PEAS = 1 indicates the UTest NVM address space is active for program or erase (if applicable) operations. The value in PEAS is captured and held when the UTest NVM block is enabled with the first interlock write done for program operations(and erase operations if the flash is unsealed). The value of PEAS is retained between sampling events (in other words, subsequent first interlock writes). The value in PEAS may be changed during erase-suspended program, and reverts back to its original state once the erase-suspended program is completed. PEAS is read only. 0 UTest NVM address space is disabled for program/erase and main address space enabled. 1 UTest NVM address space is enabled for program/erase and main address space disabled.
21 DONE	State Machine Status DONE indicates whether the embedded flash memory is performing a high voltage operation. DONE is set to a 1 on termination of the embedded flash memory reset. DONE is read only. DONE is set to a 1 at the end of program and erase high voltage sequences. **NOTE:** This bit transitions from a 0 to 1 during reset and remains at 1 after reset. 0 Flash memory is executing a high voltage operation. 1 Flash memory is not executing a high voltage operation.
22 PEG	Program/Erase Good The PEG bit indicates the completion status of the last flash memory program or erase sequence for which high voltage operations were initiated. The value of PEG is updated automatically during the program and erase high voltage operations. Aborting a program/erase high voltage operation causes PEG to be cleared, indicating the sequence failed. PEG is set to a 1 when the embedded flash memory is reset. PEG is read only. The value of PEG is valid only when PGM = 1 or ERS = 1 and after DONE transitions from 0 to 1 due to an abort or the completion of a program/erase operation. PEG is valid until PGM/ERS makes a 1 to 0 transition or EHV makes a 0 to 1 transition. The value in PEG is not valid after a 0 to 1 transition of DONE caused by PSUS or ESUS being set to logic 1. If PGM and ERS are both 1 when DONE makes a qualifying 0 to 1 transition the value of PEG indicates the completion status of the PGM sequence. This happens in an erase-suspended program operation.

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
	PEG will be 0 if an Erase-Suspended Program is attempted to a block that has been selected for Erase. PEG will also be 0 in the event of an error in the interlock write for program only. An error in the interlock write occurs when the ECC value provided to the flash memory does not match the ECC calculated within the flash memory. In the event of this, high voltage will not be applied to the array, and PEG will indicate a failed program. **NOTE:** 1. If program or erase operations are attempted on blocks that are locked, the response from embedded flash memory is PEG = 1, indicating that the operation was successful, and the contents of the block are properly protected from the program or erase operation. PEG = 1 is also true if an abort occurs during an HV request to a locked block. 2. If a program operation is attempted to a location marked as OTP, the response from the embedded flash memory is PEG = 0, indicating that the operation was not allowed, and the value interlocked was not programmed, since the desired doubleword was already programmed with a previous program operation. 0 Program or erase operation failed. 1 Program or erase operation successful.
23 PECIE	Program/Erase Complete Interrupt Enable PECIE provides a mechanism to trigger an interrupt request upon the assertion of the DONE flag due to a high voltage event (program or erase) finishing (normal, abort, suspend). If PECIE is written while not in a high voltage event, the interrupt will not immediately trigger, but will trigger after the next high voltage event is completed. 0 An interrupt request will not be generated when the DONE flag is set. 1 An interrupt request will be generated when the DONE flag is set.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 PGM	Program PGM is used to set up embedded flash memory for a program operation. A 0 to 1 transition of PGM initiates a program sequence. A 1 to 0 transition of PGM ends the program sequence. PGM can be set only under one of the following conditions: • User mode read (ERS is low and UTE is low) • Erase suspend (ERS and ESUS are 1) with EHV low PGM can be cleared only when PSUS and EHV are low and DONE is high. PGM is cleared on reset. 0 Flash memory is not executing a program sequence. 1 Flash memory is executing a program sequence.
28 PSUS	Program Suspend PSUS is used to indicate the embedded flash memory is in program suspend or in the process of entering a suspend state. The embedded flash memory is in program suspend when PSUS = 1 and DONE = 1. PSUS can be set high only when PGM and EHV are high. A 0 to 1 transition of PSUS starts the sequence which sets DONE and places the embedded flash memory in program suspend. The embedded flash memory enters suspend within Tpsus of this transition. PSUS can be cleared only when DONE and EHV are high. A 1 to 0 transition of PSUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to program. The embedded flash memory cannot exit program suspend and clear DONE while EHV is low. PSUS is cleared on reset. 0 Program sequence is not suspended. 1 Program sequence is suspended.

Table continues on the next page...

C55FMC_MCR field descriptions (continued)

Field	Description
29 ERS	Erase ERS is used to set up embedded flash memory for an erase operation. A 0 to 1 transition of ERS initiates an erase sequence. A 1 to 0 transition of ERS ends the erase sequence. ERS can only be set in user mode read (PGM is low and UTE is low). ERS can be cleared only when ESUS and EHV are low and DONE is high. ERS is cleared on reset. 0 Flash memory is not executing an erase sequence. 1 Flash memory is executing an erase sequence.
30 ESUS	Erase Suspend ESUS is used to indicate that the embedded flash memory is in erase suspend or in the process of entering a suspend state. The embedded flash memory is in erase suspend when ESUS = 1 and DONE = 1. ESUS can be set high only when ERS and EHV are high and PGM is low. A 0 to 1 transition of ESUS starts the sequence which sets DONE and places the flash memory in erase suspend. The embedded flash memory enters suspend within Tesus of this transition. ESUS can be cleared only when DONE and EHV are high and PGM is low. A 1 to 0 transition of ESUS with EHV = 1 starts the sequence which clears DONE and returns the embedded flash memory to erase. The embedded flash memory cannot exit erase suspend and clear DONE while EHV is low. ESUS is cleared on reset. 0 Erase sequence is not suspended. 1 Erase sequence is suspended.
31 EHV	Enable High Voltage The EHV bit enables the embedded flash memory for a high voltage program/erase operation. EHV is cleared on reset. EHV must be set after an interlock write to start a program/erase sequence. EHV may be set, initiating a program/erase, after an interlock under one of the following conditions: • Erase (ERS = 1, ESUS = 0) • Program (ERS = 0, ESUS = 0, PGM = 1, PSUS = 0) • Erase-suspended program (ERS = 1, ESUS = 1, PGM = 1, PSUS = 0) If a program operation is to be initiated while an erase is suspended, EHV must be cleared while in erase suspend before setting PGM. In normal operation, a 1 to 0 transition of EHV with DONE high, PSUS low, and ESUS low terminates the current program/erase high-voltage operation. In an erase-suspended program operation, a 1 to 0 transition of EHV with DONE high, ESUS high, PGM high, ERS high, and PSUS low terminates the program that is the current high voltage operation. When an operation is aborted, there is a 1 to 0 transition of EHV with DONE low and the suspend bit for the current program/erase sequence low. An abort causes the value of PEG to be cleared, indicating a failed program/erase; address locations being operated on by the aborted operation contain indeterminate data after an abort. EHV may not be written to a 1 after an abort is requested (EHV being cleared) and before DONE transitions high. A suspended operation cannot be aborted. EHV may be written during suspend. EHV must be high for the embedded flash memory to exit suspend. EHV may not be written after a suspend bit is set high and before DONE transitions high. EHV may not be set low after the current suspend bit is set low and before DONE transitions low. **NOTE:** Aborting a high voltage operation leaves FC addresses in an indeterminate data state. This may be recovered by executing an erase on the affected blocks. 0 Flash memory is not enabled to perform a high voltage operation. 1 Flash memory is enabled to perform a high voltage operation.

72.5.2 Extended Module Configuration Register (C55FMC_MCRE)

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HT	0		n256K					n64Kh			n32Kh		n16Kh		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	n64Km			n32Km		n16Km			n64Kl			n32Kl		n16Kl		
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_MCRE field descriptions

Field	Description
0 HT	High Temperature Enabled. HT provides configuration information of the module. If the flash is tuned for high temperature operation (165 °C) at reduced performance (slower read at 165 °C and less and slower program and erase at 150 °C or less), and limited features at this temperature (no margin reads), the HT status bit will be a 1. Default is 150 °C as maximum operating temperature, and the HT status bit will be a 0. HT is read only. 0 150 °C maximum operating temperature. **NOTE:** 165 °C operating temperature maximum is not supported on this device. 1 Reserved
1–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
3–7 n256K	Number of 256 KB blocks. The value of this field is dependent upon the size of the embedded flash memory. This field is read only. 00000 Zero 256 KB blocks. 00001 Two 256 KB blocks (512 KB total). 00010 Four 256 KB blocks (1 MB total). 00011 Six 256 KB blocks (1.5 MB total). 00100 Eight 256 KB blocks (2 MB total).

Table continues on the next page...

C55FMC_MCRE field descriptions (continued)

Field	Description
	00101 Ten 256 KB blocks (2.5 MB total). 00110 Twelve 256 KB blocks (3 MB total). 00111 Fourteen 256 KB blocks (3.5 MB total). 01000 Sixteen 256 KB blocks (4 MB total). 01001 Eighteen 256 KB blocks (4.5 MB total). 01010 Twenty 256 KB blocks (5 MB total). 01011 Twenty-two 256 KB blocks (5.5 MB total). 01100 Twenty-four 256 KB blocks (6 MB total). 01101 Twenty-six 256 KB blocks (6.5 MB total). 01110 Twenty-eight 256 KB blocks (7 MB total). 01111 Thirty 256 KB blocks (7.5 MB total). 10000 Thirty-two 256 KB blocks (8 MB total). 10001 Reserved 10010 Reserved 10011 Reserved 10100 Reserved 10101 Reserved 10110 Reserved 10111 Reserved 11000 Forty-eight 256 KB blocks (12 MB total). 11001 Reserved. 11010 Reserved. 11011 Reserved. 11100 Reserved. 11101 Reserved. 11110 Reserved. 11111 Reserved.
8–10 n64Kh	Number of 64 KB blocks in partitions 4 and 5. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Twelve 64 KB blocks.. 110 Sixteen 64 KB blocks. 111 Reserved.
11–12 n32Kh	Number of 32 KB blocks in partitions 4 and 5. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
13–15 n16Kh	Number of 16 KB blocks in partitions 4 and 5. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks.

Table continues on the next page...

C55FMC_MCRE field descriptions (continued)

Field	Description
	010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
16–18 n64Km	Number of 64 KB blocks in partitions 2 and 3. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
19–20 n32Km	Number of 32 KB blocks in partitions 2 and 3. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.
21–23 n16Km	Number of 16 KB blocks in partitions 2 and 3. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Eight 16 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
24–26 n64Kl	Number of 64 KB blocks in partitions 0 and 1. This field is read only. 000 Zero 64 KB blocks. 001 Two 64 KB blocks. 010 Four 64 KB blocks. 011 Six 64 KB blocks. 100 Eight 64 KB blocks. 101 Reserved. 110 Reserved. 111 Reserved.
27–28 n32Kl	Number of 32 KB blocks in partitions 0 and 1. This field is read only. 00 Zero 32 KB blocks. 01 Two 32 KB blocks. 10 Four 32 KB blocks. 11 Reserved.

Table continues on the next page...

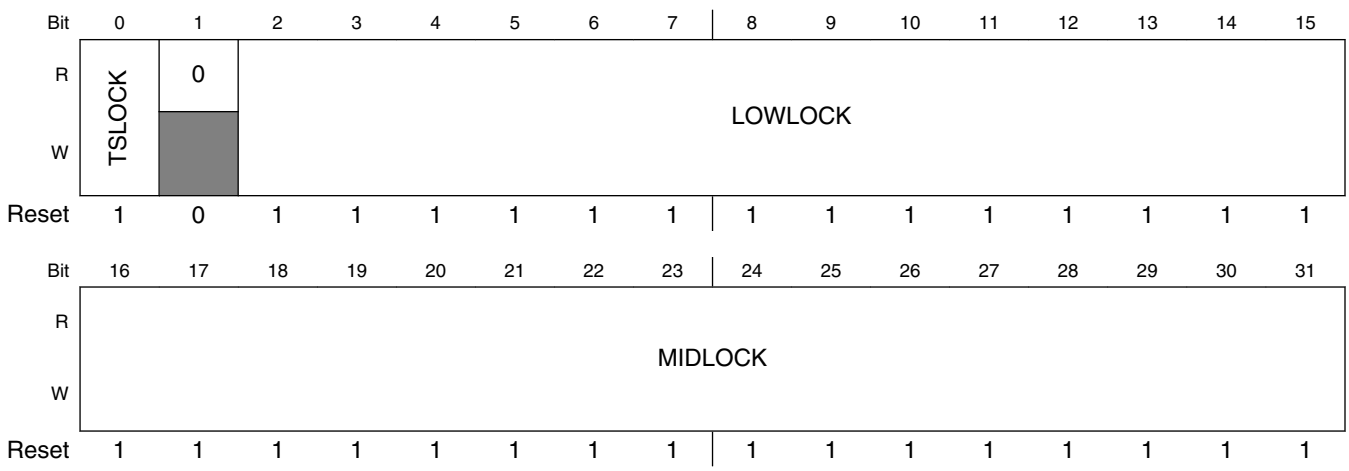
C55FMC_MCRE field descriptions (continued)

Field	Description
29–31 n16Kl	Number of 16 KB blocks in partitions 0 and 1. This field is read only. 000 Zero 16 KB blocks. 001 Two 16 KB blocks. 010 Four 16 KB blocks. 011 Six 16 KB blocks. 100 Reserved. 101 Reserved. 110 Reserved. 111 Reserved.

72.5.3 Lock 0 register (C55FMC_LOCK0)

The Lock 0 (LOCK0) register provides a means to protect blocks from being modified.

Address: 0h base + 10h offset = 10h



C55FMC_LOCK0 field descriptions

Field	Description
0 TSLOCK	UTest NVM Lock. This bit is used to lock the UTest NVM block from programs (erase protection not needed since UTest NVM is OTP and not erasable). A value of 1 in the TSLOCK register signifies that the UTest NVM block is locked for program. A value of 0 in the TSLOCK register signifies that the UTest NVM block is available to receive program pulses. The TSLOCK register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, TSLOCK register is not writable if a high voltage operation is suspended.

Table continues on the next page...

C55FMC_LOCK0 field descriptions (continued)

Field	Description
	0 UTest NVM block is available to be programmed. 1 UTest NVM block is locked and cannot be programmed.
1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWLOCK	Low Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Low Blocks starts at LOWLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations
16–31 MIDLOCK	Mid Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the Mid Blocks starts at MIDLOCK[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

72.5.4 Lock 1 register (C55FMC_LOCK1)

The Lock 1 (LOCK1) register provides a means to protect blocks from being modified.

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHLOCK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

C55FMC_LOCK1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHLOCK	High Block Lock A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the High Blocks starts at HIGHLOCK[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

72.5.5 Lock 2 register (C55FMC_LOCK2)

The Lock 2 (LOCK2) register provides a means to protect blocks from being modified.

Address: 0h base + 18h offset = 18h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KLOCK																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

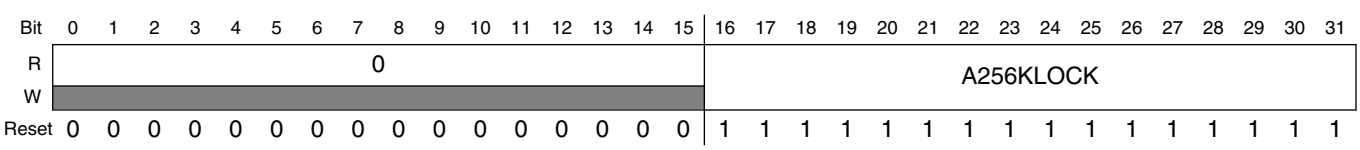
C55FMC_LOCK2 field descriptions

Field	Description
0–31 A256KLOCK	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

72.5.6 Lock 3 register (C55FMC_LOCK3)

The Lock 3 (LOCK3) register provides a means to protect blocks from being modified.

Address: 0h base + 1Ch offset = 1Ch



C55FMC_LOCK3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 A256KLOCK	256 KB Block Lock 256KLOCK has the same characteristics as LOWLOCK. Please see that description for more information. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase. A value of 0 in the lock register signifies that the corresponding block is available to receive program and erase pulses. The block numbering for the 256 KB blocks starts with 256KLOCK[0] and continues until all blocks are accounted.

Table continues on the next page...

C55FMC_LOCK3 field descriptions (continued)

Field	Description
	The lock register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation. Likewise, the lock register is not writable if a high-voltage operation is suspended. The default value of the LOCK bits is locked. In the event that blocks are not present (due to configuration or total memory size), the LOCK bits default to be locked, and are not writable. The reset value is always 1, and register writes have no effect. 0 Block is available for program and erase operations. 1 Block is locked and unavailable for program and erase operations

72.5.7 Select 0 register (C55FMC_SEL0)

The Select 0 (SEL0) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 38h offset = 38h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		LOWSEL														MIDSEL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_SEL0 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWSEL	LOW Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Low Blocks starts at LOWSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Low Blocks exist in partitions 0 and 1. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. LOWSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

Table continues on the next page...

C55FMC_SEL0 field descriptions (continued)

Field	Description
16–31 MIDSEL	Mid Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the Mid Blocks starts at MIDLSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. Mid Blocks exist in partitions 2 and 3. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. MIDSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

72.5.8 Select 1 register (C55FMC_SEL1)

The Select 1 (SEL1) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 3Ch offset = 3Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHSEL															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

C55FMC_SEL1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHSEL	High Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the High Blocks starts at HIGHSEL[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. HIGH Blocks exist in partitions 4 and 5.

Table continues on the next page...

C55FMC_SEL1 field descriptions (continued)

Field	Description
	The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. HIGHSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected for erase 1 The block is selected for erase

72.5.9 Select 2 register (C55FMC_SEL2)

The Select 2 (SEL2) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 40h offset = 40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	A256KSEL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

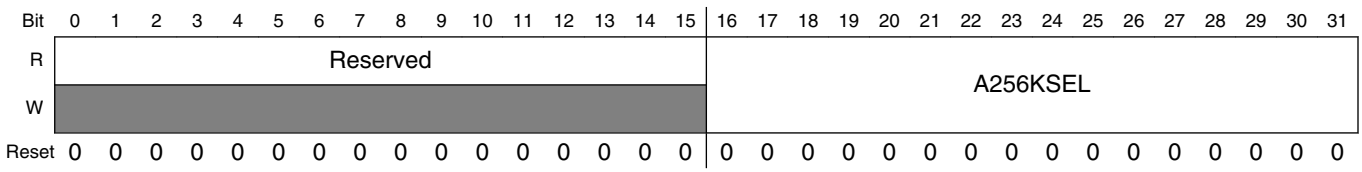
C55FMC_SEL2 field descriptions

Field	Description
0–31 A256KSEL	256 KB Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 0 The block is not selected 1 The block is selected for erase

72.5.10 Select 3 register (C55FMC_SEL3)

The Select 3 (SEL3) register provides a means to select blocks to be operated on during erase.

Address: 0h base + 44h offset = 44h



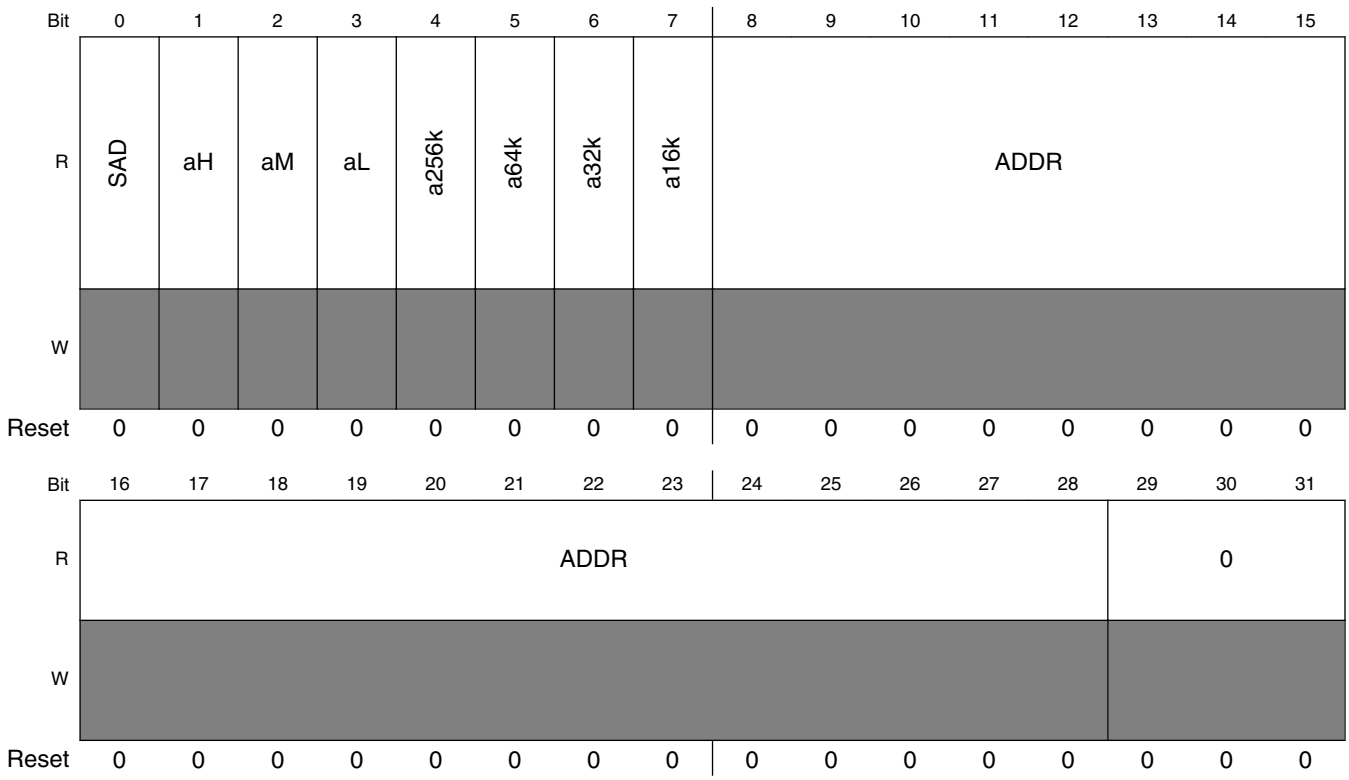
C55FMC_SEL3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. Reserved, reset to 0.
16–31 A256KSEL	256 KB Block Select. A value of 1 in the select register signifies that the block is selected for erase. A value of 0 in the select register signifies that the block is not selected. The reset value for the select registers is 0, or unselected. The block numbering for the 256 KB blocks starts with 256KSEL[0] and continues until all blocks are accounted. The blocks must be selected (or unselected) before doing an erase interlock write as part of the erase sequence. The select register is not writable once an interlock write is completed until DONE is set at the completion of the requested operation, or if a high-voltage operation is suspended. 256KSEL is also not writeable during UTest operations, when AIE is high. In the event that blocks are not present (due to configuration or total memory size), the corresponding select bits default to unselected, and are not writable. The reset value is always 0, and register writes have no effect. 1 The block is selected for erase. 0 The block is not selected.

72.5.11 Address register (C55FMC_ADR)

The Address register (ADR) provides the first failing address in the event of a failure (ECC or PGM/Erase state machine), as well as single bit correction information.

Address: 0h base + 50h offset = 50h



C55FMC_ADR field descriptions

Field	Description
0 SAD	UTest NVM Address. The SAD bit qualifies the address captured during an ECC Event Error, Single Bit Correction, or State Machine operation. SAD has the same characteristics as ADDR related to capturing of addresses. The SAD field is not writable. 0 Address captured is from main array Space. 1 Address captured is from UTest NVM array space.
1 aH	Address High region. The aH bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aH is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from high address region. 1 Address captured or to be accessed is from high address region.
2 aM	Address Mid region. The aM bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aM is set, then the ADDR field maps to that region. See ADDR description for more information.

Table continues on the next page...

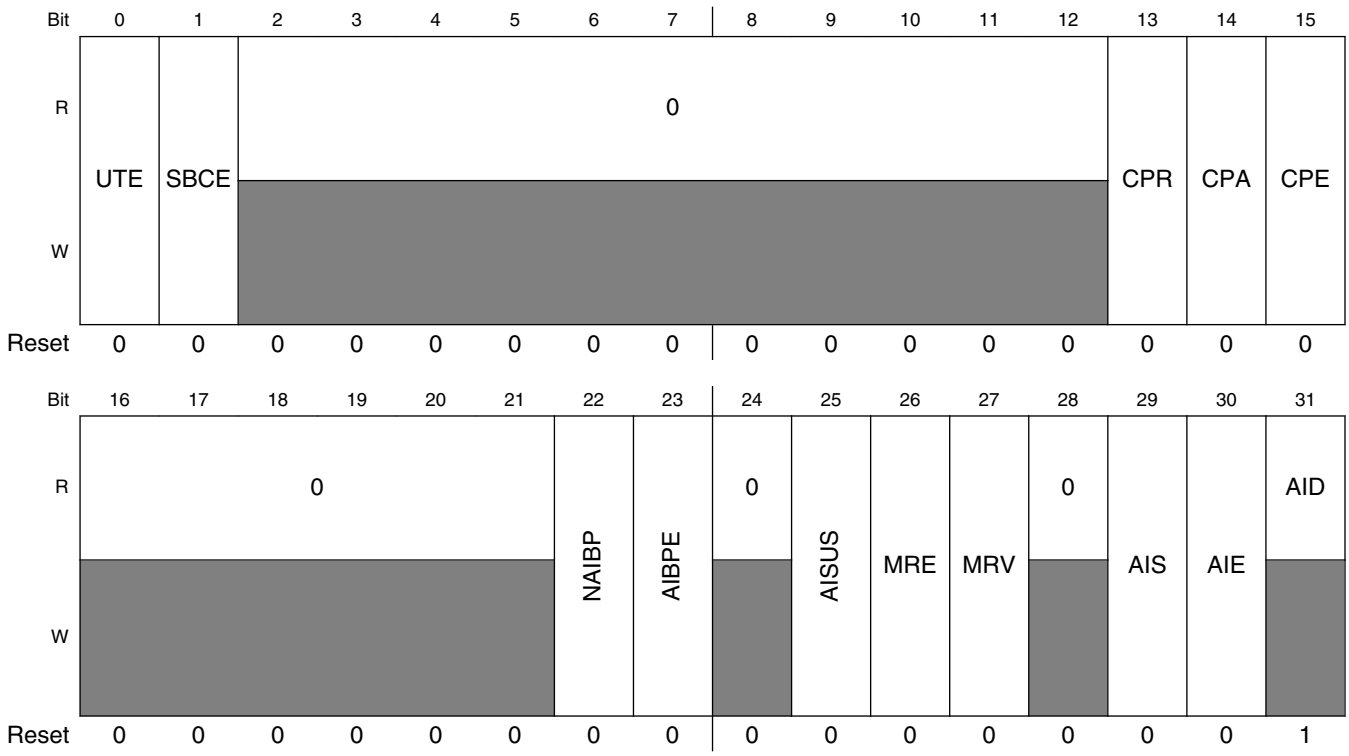
C55FMC_ADR field descriptions (continued)

Field	Description
	0 Address captured or to be accessed is not from mid address region. 1 Address captured or to be accessed is from mid address region.
3 aL	Address Low region. The aL bit qualifies the address field (ADDR) to the region of a16K, a32K and a64k. If aL is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from low address region. 1 Address captured or to be accessed is from low address region.
4 a256k	Address 256 KB region. The a256k bit qualifies the address field (ADDR) to the region. If a256k is set, then the ADDR field maps to that region. See ADDR description for more information. The a256k register is writable. 0 Address captured or to be accessed is not from 256 KB region. 1 Address captured or to be accessed is from 256 KB region.
5 a64k	Address 64 KB region. The a64k bit qualifies the address field (ADDR) to the region. If a64k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 64 KB region. 1 Address captured or to be accessed is from 64 KB region.
6 a32k	Address 32 KB region. The a32k bit qualifies the address field (ADDR) to the region. If a32k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 32 KB region. 1 Address captured or to be accessed is from 32 KB region.
7 a16k	Address 16 KB region. The a16k bit qualifies the address field (ADDR) to the region. If a16k is set, then the ADDR field maps to that region. See ADDR description for more information. 0 Address captured or to be accessed is not from 16 KB region. 1 Address captured or to be accessed is from 16 KB region.
8–28 ADDR	Address. The ADR provides the first failing address in the event of ECC event error (EER set), Single Bit Correction (SBC set), as well as providing the address of a failure that may have occurred in a state machine operation (PEG cleared). ECC event errors take priority over Single Bit Corrections, which take priority over state machine errors. This is especially valuable in the event of an RWW operation, where the read senses an ECC error or Single Bit correction, and the state machine fails simultaneously. The failing address for ECC event error is held until EER is cleared. The failing address for Single Bit Correction is held until an ECC event error, or until SBC is cleared. State machine address is held until an ECC event error or Single Bit Correction event occurs, or until the next state machine event (PEG cleared) occurs. This address is always a doubleword address that selects 64 bits. ADDR[23:3] is an offset from a base address of 0x0 for each block size region. The aH, aM, aL, a16k, a32k, a64k, and a256k qualify the block size region the ADDR field.
29–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

72.5.12 UTest 0 register (C55FMC_UT0)

The User Test 0 (UT0) register provides a means to control UTest. UTest mode gives the ability to perform test features on the flash memory. This register is only writable when the flash memory is put into UTest mode by writing a passcode.

Address: 0h base + 54h offset = 54h



C55FMC_UT0 field descriptions

Field	Description
0 UTE	U-Test Enable. This status bit gives indication when U-Test is enabled. All bits in UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 are locked when this bit is 0. This bit is not writeable to a 1, but may be cleared. The reset value is 0. The method to set this bit is to provide a password, and if the password matches, the UTE bit is set to reflect the status of enabled, and is enabled until it is cleared by a register write. The UTE password will only be accepted if PGM = 0 and ERS = 0 (program and erase are not being requested). UTE can only be cleared if AID = 1, MRE and AIE = 0. While clearing UTE, writes to set AIE or MRE will be ignored. For UTE, the password 0xF9F9_9999 must be written to the UT0 register. 0 All bits in the UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 registers are locked 1 Bits in the UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8 and UM9 registers are unlocked
1 SBCE	Single Bit Correction Enable. SBCE enables Single Bit Correction results to be observed in SBC. ECC corrections that occur when SBCE is cleared will not be logged.

Table continues on the next page...

C55FMC_UT0 field descriptions (continued)

Field	Description
	0 Single Bit Corrections observation is disabled. 1 Single Bit Correction observation is enabled.
2–12 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 CPR	Customer Programmable Read Voltage and Reference Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Read Voltage and Reference Error detection will be noted. This bit may not be set while CPE and CPA are set. If CPE or CPA are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Read Voltage and Reference Detection Error is not enabled 1 Customer Programmable Read Voltage and Reference Detection Error is enabled
14 CPA	Customer Programmable Address Encode Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a Address Encode Error detection will be noted. This bit may not be set while CPE and CPR are set. If CPE or CPR are high, this bit will be locked. If attempts are made to write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable Address Encode Detection Error is not enabled 1 Customer Programmable Address Encode Detection Error is enabled
15 CPE	Customer Programmable EDC after ECC Detection. This bit is used to control the error generation for the Customer Programmable Detection area in the UTest block. If this bit is set, and the Customer Programmable Detection location in UTest is read, a EDC after ECC Error detection will be noted. This bit may not be set while CPR and CPA are set. If CPR or CPA are high, this bit will be locked. If attempts are made o write CPR, CPA or CPE simultaneously, none of them will be written. 0 Customer Programmable EDC after ECC Detection Error is not enabled 1 Customer Programmable EDC after ECC Detection Error is enabled
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 NAIBP	Next Array Integrity Break Point. If AIBPE is set, NAIBP will be set once a single bit correction (if enabled) or double bit detection is noted during the Array Integrity test. NAIBP is not writable to 1, but may be cleared to 0. Clearing NAIBP to 0, will enable the Array Integrity operation to be re-started after a breakpoint is encountered. If the Array Integrity operation completes without encountering another correction or detection, AID will be set with NAIBP remaining 0. 1 Array Integrity state machine is at a break point. 0 Array Integrity state machine is not currently at a break point.
23 AIBPE	Array Integrity Break Point Enable. If you want to enable breakpoints during an array integrity test, AIBPE may be set. See NAIBP description for more information about array integrity breakpoints. 1 Array Integrity breakpoints are enabled during Array Integrity Checks. 0 Array Integrity breakpoints are not enabled.
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 AISUS	Array Integrity Suspend. AISUS enables a suspend of an Array Integrity Operation. Array Integrity may be suspend by Setting AISUS, and resumed by clearing AISUS. AISUS is writeable only when AID is low. AISUS is clearable only when AID is high. Attempting to write AISUS and AIE on the same clock cycle will result in only AIE getting written.

Table continues on the next page...

C55FMC_UT0 field descriptions (continued)

Field	Description
	1 Array integrity sequence is suspended. 0 Array integrity sequence not suspended.
26 MRE	Margin Read Enable. MRE combined with MRV enables user margin reads to be done. Normal user reads are not affected by MRE, although user reads while the margin read operation is ongoing are not supported. MRE is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 1 Margin reads are enabled. 0 Margin reads are not enabled.
27 MRV	Margin Read Value. MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). In order for this value to be valid, MRE must also be set. MRV is not writable if AID is low, or if AISUS is high, or if NAIBP is high. 1 One's margin reads are requested. 0 Zero's margin reads are requested.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 AIS	Array Integrity Sequence. AIS determines the address sequence to be used during array integrity checks. The default sequence (AIS = 0) is meant to replicate sequences that normal user code follows, and thoroughly checks the read propagation paths. This sequence is proprietary. The alternative sequence (AIS = 1) is logically sequential. It should be noted that the time to run a sequential sequence is shorter than the time to run the proprietary sequence. AIS is not writeable if AIE is high. **NOTE:** Only sequential addressing is supported for margin read checks. 1 Array integrity sequence is sequential. 0 Array integrity sequence is proprietary sequence.
30 AIE	Array Integrity Enable. AIE set to one starts the array integrity check done on all selected blocks. The address sequence is determined by AIS, and the MISR (UM0 through UM9) can be checked after the operation is complete, to determine if a correct signature has been obtained. Once an array integrity operation is requested (AIE=1), it may be terminated by clearing AIE if the operation has finished (AID = 1) or aborted by clearing AIE if the operation is ongoing (AID = 0). AIE is not simultaneously writable to a 1 as UTE is being cleared to a 0. 0 Array integrity checks are not enabled. 1 Array integrity checks are enabled.
31 AID	Array Integrity Done. AID is cleared upon an array integrity check being enabled (to signify the operation is ongoing). Once completed, AID is set to indicate that the array integrity check is complete. At this time the MISR (UMR registers) can be checked. AID may also assert if breakpoints are enabled (AIBPE is set), an abort is requested or a suspend is requested. AID cannot be written, and is status only. 0 Array integrity check is ongoing. 1 Array integrity check is done.

72.5.13 UMISR register (C55FMC_UM*n*)

The Multiple Input Signature registers provide a means to evaluate array integrity.

Address: 0h base + 58h offset + (4d × i), where i=0d to 8d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MISR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

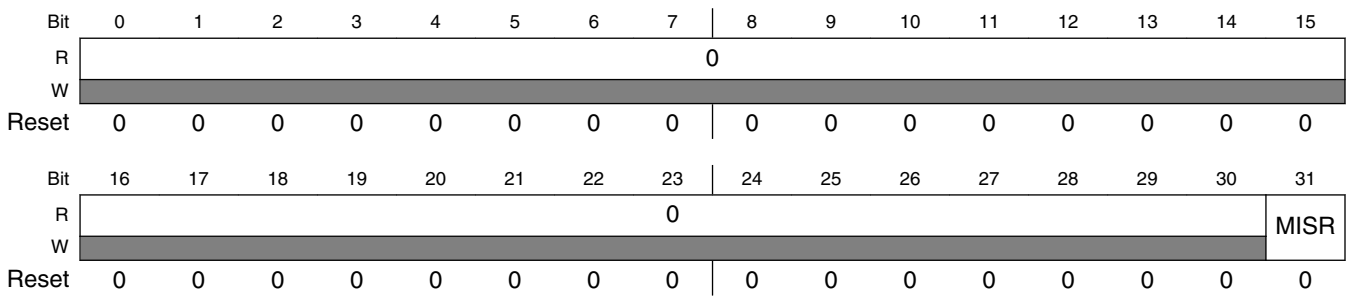
C55FMC_UM*n* field descriptions

Field	Description
0–31 MISR	MISR[31:0]. The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. **NOTE:** Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. The MISR register provides a means to calculate a MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^{7} + x^{6} + x^{5} + x^{4} + x^{2} + 1$ The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read. The MISR register is used in array integrity operations. If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained.

72.5.14 UMISR register (C55FMC_UM9)

The Multiple Input Signature registers provide a means to evaluate array integrity.

Address: 0h base + 7Ch offset = 7Ch



C55FMC_UM9 field descriptions

Field	Description
0–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 MISR	MISR[288]. The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required). The MISR can be seeded to any value by writing the MISR registers. **NOTE:** Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. The MISR register provides a means to calculate a MISR during array integrity operations. The MISR can be represented by the following polynomial: $x^{289} + x^{7} + x^{6} + x^{5} + x^{4} + x^{2} + 1$ The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read. The MISR register is used in array integrity operations. If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers is not recalculated, and the previous value is retained

72.5.15 Over-Program Protection 0 register (C55FMC_OPP0)

The Over-Program Protection 0 (OPP0) register provides a means to protect blocks from being over-programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 80h offset = 80h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0		LOWOPP														MIDOPP															
W																																
Reset	0	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- MIDOPP field: The reset value of the MIDOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.
- LOWOPP field: The reset value of the LOWOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP0 field descriptions

Field	Description
0–1 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2–15 LOWOPP	Low Block Over-Program Protection[13:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Low Blocks starts at LOWOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The LOWOPP register is not writable, and is status only. The default value of the LOWOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.
16–31 MIDOPP	Mid Block Over-Program Protection[15:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the Mid Blocks starts at MIDOPP[0] with 16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted.

Table continues on the next page...

C55FMC_OPP0 field descriptions (continued)

Field	Description
	The MIDOPP register is not writable, and is status only. The default value of the MIDOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

72.5.16 Over-Program Protection 1 register (C55FMC_OPP1)

The Over-Program Protection 1 (OPP1) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 84h offset = 84h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHOPP															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- HIGHOPP field: The reset value of the HIGHOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP1 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHOPP	High Block Over-Program Protection[15:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the High Blocks starts at HIGHOPP[0] with16 KB block region (if available), then the 32 KB block region (if available), and lastly the 64 KB block region (if available). This continues until all blocks are accounted. The HIGHOPP register is not writable, and is status only. The default value of the HIGHOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

72.5.17 Over-Program Protection 2 register (C55FMC_OPP2)

The Over-Program Protection 2 (OPP2) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 88h offset = 88h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	A256KOPP																															
W																																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:

- A256KOPP field: The reset value of the A256KOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP2 field descriptions

Field	Description
0–31 A256KOPP	256K Block Over-Program Protection[31:0]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at A256KOPP[0] and continues until all blocks are accounted. The 256KOPP register is not writable, and is status only. The default value of the A256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 0 The corresponding block is available to be over-programmed.

72.5.18 Over-Program Protection 3 register (C55FMC_OPP3)

The Over-Program Protection 3 (OPP3) register provides a means to protect blocks from being over programmed. This register shows the over-program protection status.

NOTE

The reset value of this register is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

Address: 0h base + 8Ch offset = 8Ch

Bit	0–15	16–31
R	0	A256KOPP
W		
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0* 0*

* Notes:
- A256KOPP field: The reset value of the A256KOPP field is set at the factory and varies among devices. See the chip-specific C55FMC information for details.

C55FMC_OPP3 field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 A256KOPP	256K Block Over-Program Protection[47:32]. A value of 1 in a bit of the over-program protection register signifies that the corresponding block is protected from over-programming. A value of 0 in the OPP register signifies that the corresponding block is available to be over-programmed. The block numbering for the 256 KB Blocks starts at 256KOPP[0] and continues until all blocks are accounted. The 256KOPP field is not writable, and is status only. The default value of the A256KOPP bits is not OPP. In the event that blocks are not present (due to configuration or total memory size), the OPP bits default not OPP, and will remain 0. 1 The corresponding block is protected from over-programming. 0 The corresponding block is available to be over-programmed.

72.5.19 Test Mode Disable Password Check register (C55FMC_TMD)

The Test Mode Disable Password Check (TMD) register provides a means to provide a challenge password to disable the Test Mode Disable Seal block select.

Writes to the register have no effect except for a password challenge. Reads to this register always return 0.

Address: 0h base + 90h offset = 90h

Bit	0–15	16–31
R W	PWD	
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

C55FMC_TMD field descriptions

Field	Description
0–31 PWD	Password challenge

72.6 Functional Description

The embedded flash memory consists of address spaces in four groupings:

- Low address space
- Mid address space
- High address space
- 256 KB address space

The main address space is divided into partitions. Each address space mentioned above consists of a partition pair. Partitions are used to determine locations for valid read-while-write (RWW) operations. While the embedded flash memory is performing a write (program or erase) to a given partition, it can simultaneously perform a read from any other partition. For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written). For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.

The main address space is also divided into blocks to implement independent erase and program protection. The UTest NVM block also exists as a block, and has independent program protection. The UTest NVM block is included to support systems that require nonvolatile memory (NVM) for security or to store system initialization information.

A number of MCR bits are protected against write when another bit, or set of bits, is in a specific state. These write locks are covered on a bit-by-bit basis in Module Configuration Register (C55FMC_MCR). The write locks do not consider the effects of trying to write two or more bits simultaneously. The embedded flash memory does not allow bits to be written simultaneously which put the device into an illegal state. This is implemented through a priority mechanism among the bits. The bit changing priorities are detailed in the following table.

Table 72-2. MCR bit set/clear priority levels

Priority level	MCR Bit(s)
1	ERS
2	PGM
3	EHV
4	ESUS, PSUS

If two or more MCR bits are written simultaneously only the bit with the lowest priority number is accepted for modification. Setting two bits with the same priority number is prevented by existing write locks or do not put the flash memory in an illegal state.

For example, setting ERS and PGM simultaneously results in only ERS being set. Attempting to clear EHV while setting PSUS results in EHV being cleared, while PSUS is unaffected.

Each read of the embedded flash memory retrieves a page, or eight consecutive words (256 bits), of information. The address for each word retrieved within a page differs from the other addresses in the page only by address bits [4:2]. The flash memory page read architecture easily supports both cache and burst mode at the BIU level for high-speed read applications.

The embedded flash memory supports fault tolerance through error correction code (ECC) and error detection. ECC implemented within the embedded flash memory corrects single-bit failures and detects double-bit failures.

A software mechanism is provided to independently lock and unlock each block.

Program and erase of the embedded flash memory requires multiple system clock cycles to complete. The program and erase sequence may be suspended or aborted.

The embedded flash memory may reside in various modes. The modes that are available include:

- User mode
- Low Power mode
- UTest mode

Each of these modes is discussed in detail in the following sections.

72.6.1 User Mode

In user mode the embedded flash memory may be read and written (register writes and interlock writes), programmed or erased. The following sub-sections define all actions that may be performed in user mode.

72.6.1.1 Read and write

The default state of the embedded flash memory is read. The main and UTest NVM address space can be read only in the read state. The MCR is always available for read, except when the embedded flash memory is in low power mode. The embedded flash memory enters the read state on reset. The embedded flash memory is in the read state under four sets of conditions:

- The read state is active when the embedded flash memory is enabled (User Mode Read).
- The read state is active when MCR[PGM] or MCR[ERS] are high and high-voltage operation is ongoing (read-while-write).

NOTE

Reads done to the partition(s) being operated on (either erased or programmed) result in an error and the MCR[RWE] bit is set.

- The read state is active when MCR[PGM] and MCR[PSUS] are high (program suspend).
- The read state is active when MCR[ERS] and MCR[ESUS] are high and MCR[PGM] is low (erase suspend).

In embedded flash memory, flash core reads return 256 bits (1 page). Register reads return 32 bits of data.

NOTE

Flash core reads are done through the BIU. In many cases the BIU does read page buffering to allow sequential reads to be done with higher performance. This could provide a data coherency issue that must be handled with software. Data coherency may be an issue after a program or erase operation, as well as UTest NVM block operations.

In user mode, registers may be written. Array may be written to do interlock writes.

Register reads to unmapped register address space return all zeroes.

Register writes to unmapped register address space have no effect.

Interlock writes that are attempted during a high-voltage operation (MCR[EHV] = 1 or MCR[DONE] =0) will result in the interlock write being ignored, and address and data will not be updated.

NOTE

Care must be taken when doing 32-bit interlock writes. Performing two 32-bit interlock writes to the same word location is not supported. In addition, mixing 32-bit interlock writes with 64-bit interlock writes to the same double-word location is not supported. If these combinations are done, and a program operation is executed, the result will be MCR[PEG] = 0. See the Module Configuration Register (MCR) section for more information.

72.6.1.2 Program

A flash memory program sequence operates on any page within the flash core. Within a page, up to eight words may be altered in a single program operation. Also, up to four pages can be altered in a single program operation. Whenever the array is programmed, the ECC bits also get programmed. ECC is handled on a 64-bit boundary. Thus, if only one word in any given 64-bit ECC segment is programmed, the adjoining word (in that segment) should not be programmed, since ECC calculation has already completed for that 64-bit segment. Attempts to program the adjoining word will probably result in an operation failure. It is recommended that all programming operations be from 64 bits to 1024 bits, and be 64-bit aligned. The programming operation should completely fill selected ECC segments within the page. Only one program is allowed per 64-bit ECC segment between erases.

Warning

In rare cases over-programming of a 64-bit ECC segment may be done (EEPROM emulation).

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1.

NOTE

If a logic 0 is attempted to be over-programmed by a logic 1, the resulting operation will fail (MCR[PEG] = 0), and the 0's that are interlocked will be merged (ORed) with 0's that are already present in the 64-bit ECC segment, unless the block is designated as an Over-Program Protected block.

Addresses in locked/disabled blocks cannot be programmed. Values may be programmed in any or all of eight words, within a page, with a single program sequence. Page-bound words have addresses which differ only in address bits [4:2]. Up to four pages can be programmed at once on a quad-page boundary, which differ only in address bits [6:5]. The program operation consists of the following sequence of events:

1. Change the value in the MCR[PGM] bit from a 0 to a 1.

   NOTE

   Ensure the block that contains the address to be programmed is unlocked.

2. Write the first address to be programmed with the program data. The embedded flash memory latches address bits [22:7] and SoC-specific UTest NVM enable at this time. The embedded flash memory latches data written as well. This write is referred to as

a program data interlock write. An interlock write may be done as a 64-bit transaction or a 32-bit transaction. Refer to the Flash Memory Controller chapter for information on the size of writes that are allowed.

3. If more than one word or doubleword is to be programmed, write each additional address in the page with data to be programmed. This is referred to as a program data write. The embedded flash memory ignores address bits [22:7] and SoC-specific UTest NVM enable for program data writes. All unwritten data words default to 0xFFFF_FFFF.
4. Write a logic 1 to the MCR[EHV] bit to start the internal program sequence or skip to step 9 to terminate.
5. Wait until the MCR[DONE] bit goes high.

NOTE

Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more addresses are to be programmed, return to step 2.
9. Write a logic 0 to the MCR[PGM] bit to terminate the program sequence.

The first write after a program is initiated determines the quad-page address to be programmed. Program may be initiated with the 0 to 1 transition of the MCR[PGM] bit or by clearing the MCR[EHV] bit at the end of a previous program. This first write is referred to as an interlock write. The interlock write determines if the UTest NVM or normal array space is to be programmed by sampling SoC-specific UTest NVM enable and causing MCR[PEAS] to be set/cleared.

In the case of an erase-suspended program, the values in MCR[PEAS] may be modified via the program interlock write, enabling erase-suspended programs to and from UTest NVM space.

An interlock write must be performed before setting MCR[EHV]. A program sequence may be terminated by clearing MCR[PGM] prior to setting MCR[EHV].

After the interlock write, additional writes affect the data to be programmed at the word location determined by address bits [4:2], as well as the page location within a 1024-bit segment (determined by address bits [6:5]). Unwritten locations default to a data value of 0xFFFF_FFFF. If multiple writes are done to the same location, the data for the last write is used in programming.

While DONE is low, EHV is high, and PSUS is low, then EHV may be cleared, resulting in a program abort. A program abort forces the embedded flash memory to step 8 of the program sequence. An aborted program results in PEG being set low, indicating a failed operation. The data space being operated on before the abort contains indeterminate data. A program sequence may not be aborted while in program suspend.

Warning

Aborting a program operation leaves the flash core addresses being programmed in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

72.6.1.2.1 Program software locking

A software mechanism is provided to independently lock/unlock blocks against program and erase.

Software locking is done through the LOCK0, LOCK1, LOCK2, and LOCK3 registers. These may be written through register writes, and may be read through register reads.

72.6.1.2.2 Program hardware locking

Hardware locking which only affects the program operation is also available for UTest, 16 KB, 32 KB, 64 KB, and 256 KB blocks.

72.6.1.2.3 Program suspend/resume

The program sequence may be suspended to allow read access to the flash core. It is not possible to erase during a program suspend, or to program during a program suspend. These operations are prevented by register locking described in the Module Configuration Register (MCR) section. Read-while-write operation may also be used to read the array during a program sequence, providing the read is to a different partition.

A program suspend can be initiated by changing the value of the MCR[PSUS] bit from a 0 to a 1. MCR[PSUS] can be set high at any time when MCR[PGM] and MCR[EHV] are high. A 0 to 1 transition of MCR[PSUS] causes the embedded flash memory to start the sequence to enter program suspend, which is a read state. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended. At this time flash core reads may be attempted. Once suspended, the flash core may only be read. During suspend, all reads to flash core locations targeted for program, and blocks targeted for erase, return indeterminate data.

The program sequence is resumed by writing a logic 0 to MCR[PSUS]. MCR[EHV] must be set to a 1 before clearing MCR[PSUS] to resume operation. When the operation resumes, the embedded flash memory continues the program sequence from one of a set of predefined points. This may extend the time required for the program operation.

NOTE

Repeated suspends at a high frequency may result in the operation not completing normally. Although suspend frequency is not limited, enabling at least 20 µs between suspend resume and future suspend requests improve the opportunity for the flash to complete the operation successfully.

72.6.1.2.4 Over-program protection enable

One-time programming can be enabled on a block basis. In an over-program protection enabled block, any doubleword which has already been programmed cannot be programmed again. The over-program protection enable does not affect erase operation. Attempts to over-program will result in MCR[PEG] being cleared. If any doubleword within the quad-page has an over-program protection violation, MCR[PEG] will be cleared, and no doublewords will be programmed.

One-time programming can be enabled for 16 KB, 32 KB, 64 KB, or 256 KB blocks.

Over-program protection is enabled by sideband signals, and if enabled, the program operation will be modified to check for programmed bits prior to doing a high-voltage program event.

72.6.1.2.5 Successful program address

To enable the SoC to create logic to authenticate operations with a program (such as in a diary operation), the flash module provides the last successful address programmed as a sideband signal. This value will be held until the next successful program. A successful program is defined as a program operation of data that is not all 1's to an unlocked block, and the operation results in MCR[PEG] returning a 1. If any doubleword within the pages being programmed is interlocked with all 1's, the sideband signal will not be updated. An address of all 1's, and each of the block size selects selected, is the default (and invalid) address after reset.

72.6.1.3 Erase

Erase changes the value stored in all bits of the selected block(s) to logic 1. An erase sequence operates on any combination of blocks in the main address space. The erase sequence is fully automated within the flash memory. Blocks to be erased must be selected prior to initiating the erase sequence. Locked/disabled blocks cannot be erased. If multiple blocks are selected for erase during an erase sequence, the blocks are erased sequentially starting with the lowest numbered block and terminating with the highest (low first, mid second, high last, 16 KB first, 32 KB second, 64 KB third, last blocks are 256 KB). The erase sequence consists of the following events:

1. Change the value in the MCR[ERS] bit from 0 to 1.
2. Select the block or blocks to be erased by writing ones to the appropriate registers in SEL0, SEL1, SEL2, or SEL3 registers.

NOTE

Lock and Select are independent. If a block is selected and locked, no erase occurs.

3. Write to any address in flash memory. This is referred to as an erase interlock write. An erase interlock write to UTest NVM space is not allowed if sealed (since erase is not allowed in the UTest NVM space when sealed).
4. Write a logic 1 to the MCR[EHV] bit to start an internal erase sequence or skip to step 9 to terminate.
5. Wait until the MCR[DONE] bit goes high.

NOTE

Since MCR[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCR[DONE] as a 0 prior to this step, depending on the operation selected.

6. Confirm MCR[PEG] = 1.
7. Write a logic 0 to the MCR[EHV] bit.
8. If more blocks are to be erased, return to step 2.
9. Write a logic 0 to the MCR[ERS] bit to terminate the erase.

After setting ERS, one write, referred to as an interlock write, must be performed before EHV can be set to a 1. Data words written during erase sequence interlock writes are ignored. The erase sequence may be terminated by clearing ERS before setting EHV.

An erase operation may be aborted by clearing EHV, assuming DONE is low, EHV is high, and ESUS is low. An erase abort forces the embedded flash memory to step eight of the erase sequence. An aborted erase results in PEG being set low, indicating a failed operation. The block(s) being operated on before the abort contain indeterminate data. An erase sequence may not be aborted while in erase suspend.

72.6.1.3.1 Erase software locking

Software Locking affect erase operation. For details on this see Program software locking.

72.6.1.3.2 Erase hardware locking

Hardware locking which only affects the erase operation is also available for UTest, 16 KB, 32 KB, 64 KB, and 256 KB blocks.

72.6.1.3.3 Erase suspend/resume

The erase sequence may be suspended to allow read access to the flash core. The erase sequence may also be suspended to program (Erase-Suspended Program) the flash core. A program started during erase suspend can in turn be suspended. Only one erase suspend and one program suspend are allowed at a time during an operation. It is not possible to erase during an erase suspend, or program during a program suspend. During suspend, all reads to flash core locations targeted for program and blocks targeted for erase return indeterminate data.

Read-while-write operation may also be used to read the array during an erase sequence, provided the read is to a partition not selected for erase.

An erase suspend can be initiated by changing the value of the MCR[ESUS] bit from a 0 to a 1. MCR[ESUS] can be set to a 1 at any time when MCR[ERS] and MCR[EHV] are high and MCR[PGM] is low. A 0 to 1 transition of MCR[ESUS] causes the embedded flash memory to start the sequence which places it in erase suspend. MCR[DONE] must become a logic level 1 before the embedded flash memory is suspended and further actions are attempted. Once suspended, the array may be read or a program sequence may be initiated (erase-suspended program). Before initiating a program sequence, MCR[EHV] must first be cleared. If a program sequence is initiated, the values of SoC-specific UTest NVM enable is recaptured. Once the erase-suspended program is completed, the value of PEAS is returned to its erase value. Flash core reads that occur while MCR[ESUS] = 1 from the block(s) being erased will return indeterminate data.

The erase sequence is resumed by writing a logic 0 to MCR[ESUS]. MCR[EHV] must be set to a 1 and MCR[PGM] must be cleared (in the event of an erase-suspended program) before MCR[ESUS] can be cleared to resume the operation. The embedded flash memory continues the erase sequence from one of a set of predefined points. This may extend the time required for the erase operation.

NOTE

1. In an erase-suspended program, programming flash core locations in blocks which were being operated on will respond by completing the operation with a fail code (MCR[PEG] = 0). Programming voltages will not be applied to the memory array in this case.
2. Repeated suspends at a high frequency may result in the operation not completing normally. Although suspend frequency is not limited, enabling at least 5ms between suspend resume and future suspend requests improve the opportunity for the flash to complete the operation successfully.

72.6.2 Low Power mode

In Low Power mode, the embedded flash memory enables VDDF power gating, and disables circuits on the VFLASH domain. No reads from or writes to the embedded flash memory are possible when in Low Power mode.

Prior to entering Low Power Mode, it is required that all program, erase, and UTest operations be stopped prior to entering the mode. If Low Power mode is entered while an operation is suspended, entering Low Power mode will have the same effect as a reset during suspend.

When in Low Power mode, register access is prevented. Flash core accesses are also prevented until power mode is exited. Flash core reads and writes may occur after power mode is exited.

The embedded flash memory returns to a post-reset state in all cases.

72.6.3 UTest mode

UTest mode is a mode into which customers can put the embedded flash memory so as to do specific tests that check the integrity of the embedded flash memory.

72.6.3.1 Array integrity self check

The Array Integrity (AI) operation is a flash-specific self-test feature in which a MISR signature value is generated from data read from selected flash blocks. After the operation is complete, customer code compares the resulting MISR signature to the expected value to determine if an incorrect read or ECC detection occurred.

Array integrity operations are fully contained within the flash module. Results of Array integrity operations are not reported beyond the boundary of the flash module.

During array integrity test, user mode array reads are ignored to ensure the array integrity operation is not corrupted. User mode array reads may be executed if suspended or at a breakpoint.

AI read operations are executed using a predefined address sequence selected via the UT0[AIS] register field value. The data to be read is customer-specific, e.g., user program code that has been programmed into the flash memory. Any random or nonrandom code is valid. The expected MISR signature is calculated based on the specific data to be read.

Note that array integrity MISR signature calculations are performed after ECC detection and correction so correctable ECC events do not cause the MISR signature to deviate from the expected value.

Array integrity processing requires the flash memory controller's PFLASH_PFCR1[RWSC] and PFLASH_PFCR1[APC] register fields to be set to values that are valid for the system frequency being used. Only the valid/recommended RWSC and APC values for a given frequency, as stated in the device data sheet, are supported.

The Array integrity check consists of the following sequence:

1. Enable UTest mode by setting the UT0[UTE] bit to 1.
2. Select one or more blocks for an array integrity check by writing 1's to the appropriate bits in the SEL0 – SEL3 registers. Blocks selected for AI test do not need to be unlocked.

   Flash blocks that have not been selected are still read as part of the array integrity sequence to enable full transition coverage. Reads on unselected blocks are not captured in the MISR. Selecting fewer blocks may not result in a faster array integrity execution time, depending on the combinations of blocks selected and the sequence.

   Blocks protected with the Test Mode Disable Seal are still read as part of the array integrity sequence. The resulting read on sealed blocks is not captured in the MISR, but the Single Bit Correction, Double Bit Detection and breakpoints are still honored.

NOTE

It is not possible to do array integrity operations on the UTest NVM block.

3. If desired, set the UT0[AIS] bit to 1 for sequential addressing only.

   For normal integrity checks of the flash memory, sequential addressing is recommended. If it is required to more fully check the read path (in a diagnostic mode), it is recommended that UT0[AIS] be left at 0, to use the address sequence that checks the read path more fully, and examine read transitions. This sequence takes more time.

4. Seed the MISR registers (UM0 — UM9) with desired values.
5. If breakpoints are desired, set UT0[AIBPE] to 1, and ensure that MCR[EER] and MCR[SBC] are cleared. If it is desired to break on single-bit correction, the UT0[SBCE] must be set.
6. Initiate AI by setting the UT0[AIE] bit to 1.

   The array integrity operation may be aborted prior to UT0[AID] going high. This may be done by clearing the UT0[AIE] bit and continuing to the next step. It should be noted that in the event of an aborted array integrity check the MISR registers contain a signature for the portion of the operation that was completed prior to the abort, and are not deterministic. Prior to doing another array integrity operation, the UM0 – UM9 registers may need to be initialized to the desired seed value by doing register writes.

   The array integrity operation may be suspended prior to UT0[AID] going high. UT0[AISUS] may be set to request an array integrity suspend. After the UT0[AISUS] bit is set, the user should wait for UT0[AID] bit to go high, which indicates the flash has entered the suspend state, and normal reads to the flash may be done. After the UT0[AID] bit goes high, the UT0[AISUS] bit may be cleared to resume the array integrity sequence.

7. Wait until the UT0[AID] bit goes high, indicating that array integrity test is complete.
8. If breakpoints are enabled, check the UT0[NAIBP] bit. If set to 1, the MCR[EER], MCR[SBC], and ADDR registers may be checked to determine the cause of the break and the address of the break. Prior to resuming the operation, clear the MCR[SBC] or MCR[EER] bits, then resume the operation by clearing the UT0[NAIBP] bit. Continue to wait until the UT0[AID] bit goes high. If breakpoints are not enabled, or if the UT0[NAIBP] bit is low when the UT0[AID] bit goes high, the operation is complete. Continue to the next step.
9. Read values in the MISR registers (UM0 — UM9) to ensure correct signature.

NOTE

Array integrity reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

10. Write a logic 0 to the UT0[AIE] bit to complete array integrity test.

72.6.3.2 User margin read

User Margin Read is a flash-specific self-test feature that enables reads to be done at a Program margin or Erase margin level.

Margin reads are performed with an internal read reference voltage that has been adjusted, i.e., "margined" toward the threshold voltage being checked. The read reference voltage is either margined toward a 0 (programmed bit) or margined towards a 1 (erased bit), depending on the margin read operation selected.

User margin read is done using the array integrity interface, and has all the associated features of the array integrity interface (MISR and Breakpoints). User margin read is a self-timed event, and is independent of system clocks or wait states selected.

Margin read operations are fully contained within the flash module. Results of margin read operations are not reported beyond the boundary of the flash module.

During margin read operations, user mode array reads are ignored to ensure the operation is not corrupted. User mode array reads may be executed if suspended or at a breakpoint.

ECC corrections and detections are noted during the user margin read test. Note that margin read MISR signature calculations are performed *after* ECC detection and correction so correctable ECC events do not cause the MISR signature to deviate from the expected value.

The data to be read is customer-specific, e.g., user program code that has been programmed into the flash memory. Any random or non-random code is valid. The expected MISR signature is calculated based on the specific data to be read. After the operation is completed the margin read results can be checked by reading the MCR[EER] and MCR[SBC] bits to determine if zero, one, or two bits are being detected by the margin read, and comparing the generated MISR signature to the expected value.

The use model for margin read is in the event of a user-detected single bit correction (through user reads). A margin read may be done to check for a possible second bit falling within the selected margin levels.

The procedure for doing margin reads is:

1. Enable UTest mode by setting the UT0[UTE] bit to 1.
2. Select one or more blocks for margin read check by writing 1's to the appropriate bits in SEL0 – SEL3 registers. Blocks selected for margin read do not need to be unlocked.

   Flash blocks that have not been selected are still read as part of the margin read sequence to enable full transition coverage. Reads on unselected blocks are not captured in the MISR. Selecting fewer blocks does not result in a faster margin read execution time.

   Blocks protected with the Test Mode Disable Seal are still read as part of the margin read sequence. The resulting read on sealed blocks is not captured in the MISR, but the Single Bit Correction, Double Bit Detection and breakpoints are still honored.

   NOTE

   It is not possible to do margin read operations on the UTest NVM block.

3. Set the UT0[AIS] bit to 1 for sequential addressing only.

   NOTE

   Only sequential addressing is supported for margin read checks.

4. Seed the MISR registers (UM0 — UM9) with desired values.
5. If breakpoints are desired, set UT0[AIBPE] to 1, and ensure that MCR[EER] and MCR[SBC] are cleared. If it is desired to break on single-bit correction, the UT0[SBCE] bit must be set to 1.
6. Initiate margin read by setting the UT0[MRE] bit to 1.
7. Select either one's margin or zero's margin by setting the UT0[MRV] bit to the appropriate value.
8. Set the UT0[AIE] bit.

   During margin read operations, with UT0[AID] low, it is not recommended to attempt write operations to the MCR[SBC] and MCR[EER] bits. It is recommended that these bits only be written during suspend, breakpoints, or at completion of the Margin Read operation.

9. Wait until the UT0[AID] bit goes high, indicating that margin read test is complete.

   If breakpoints were enabled or a suspend was requested during margin read, the operation may be at a breakpoint or a suspend state. See Step 8 in section Array integrity self check for more information.

10. Read values in the MISR registers (UM0 — UM9) to ensure correct signature.

NOTE

Margin reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISR, and do not update the MCR[EER] and MCR[SBC] error bits.

11. Write a logic 0 to the UT0[AIE] bit.

72.7 Initialization information

A reset is the highest priority operation for the embedded flash memory and terminates all other operations.

The embedded flash memory uses reset to initialize register and status bits to their default reset values. If the embedded flash memory is executing a program or erase operation (PGM or ERS = 1) and a reset is issued, the operation is aborted and the embedded flash memory disables the high-voltage logic without damage to the high-voltage circuits. Reset aborts all operations and forces the embedded flash memory into user mode ready to receive accesses.

After reset is requested, MCR[DONE] goes low, and remains low during reset and reset recovery. At the end of reset recovery, MCR[DONE] transitions from a 0 to a 1.

After reset is completed, register reads may be done, although it should be noted that registers that require updating from UTest NVM information, or other inputs, may not read updated values until MCR[DONE] transitions high.

During reset recovery, register writes are not allowed until the MCR[DONE] bit transitions high to indicate reset recovery is completed.

Warning

Resetting during a program or erase operation leaves the flash core blocks being programmed or erased in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

Chapter 73
Memory Error Management Unit (MEMU)

73.1 Chip-specific MEMU information

73.1.1 MEMU configuration

This section describes the configuration of MEMU_0 and MEMU_1 specific to the MPC5746C.

73.1.2 MEMU system connections

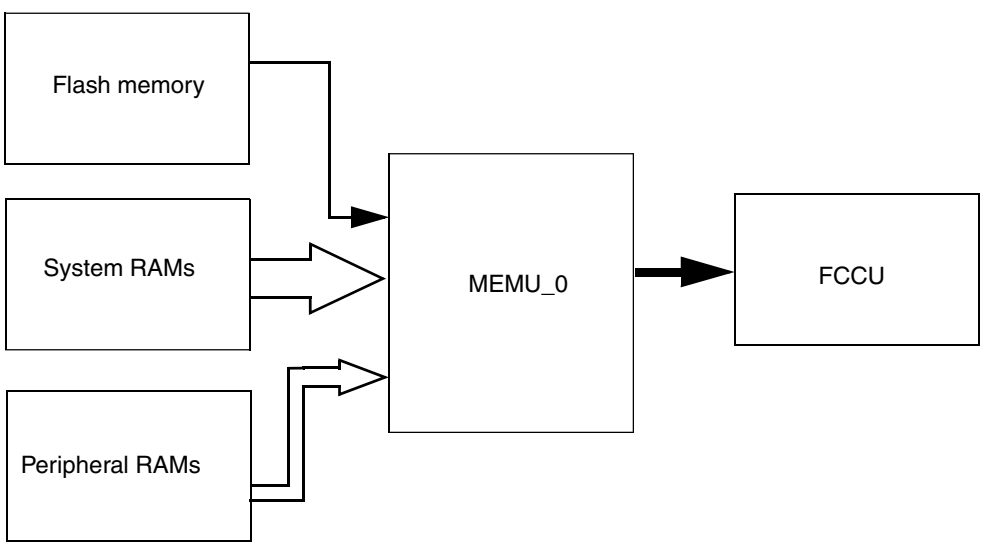


Figure 73-1. MEMU_0 system connections

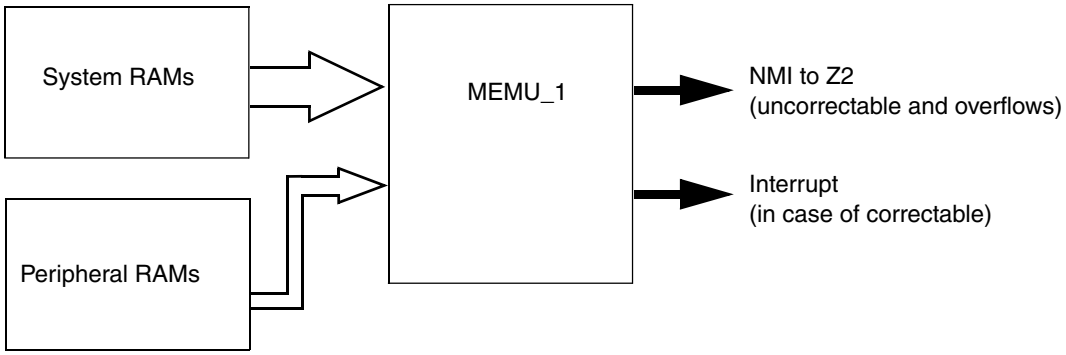


Figure 73-2. MEMU_1 system connections

73.1.3 MEMU error sources

This chip includes two instances of MEMU as follows:

- MEMU_0 (for error sources in the full power domain)
- MEMU_1 (for error sources in the LPU domain)

The following tables shows the MEMU error sources for each instance configured on this chip.

Table 73-1. MEMU_0 error sources

Instance	Value
Number of system RAM unique error sources	10
Number of peripheral RAM unique error sources	5
Number of flash memory unique error sources	3

Table 73-2. MEMU_1 error sources

Instance	Value
Number of system RAM unique error sources	4
Number of peripheral RAM unique error sources	2
Number of flash memory unique error sources	0

73.1.4 Reporting table implementation

MEMU establishes the following categories of error reporting:

- Flash memory error reporting

- System RAM error reporting (any RAM that is CPU accessible falls into this category)
- Peripheral RAM error reporting (any RAM that is not CPU accessible falls into this category)

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

- Correctable error reporting table
- Uncorrectable error reporting table

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

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

Table 73-3. Number of entries in each reporting table for MEMU_0

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

Table 73-4. Number of entries in each reporting table for MEMU_1

Error source	Number of entries in correctable error reporting table	Number of entries in un-correctable error reporting table
Flash memory	0	0
Peripheral RAM	1	1
System RAM	3	1

73.1.5 Concurrent overflow register (OFLW) implementation for MEMU_0

Errors are reported to the MEMU from these categories:

- Flash memory
- System RAM
- Peripheral RAM

In an overflow condition, it is possible to identify the unique error source that caused the overflow. Corresponding error sources for the OFLW register details the corresponding error source to the respective bit-field in the Concurrent Overflow registers.

Positions not explicitly listed in the tables below are reserved.

Table 73-5. Corresponding error sources for system RAM OFLW0 register

Bit number	Error source	Clock relationship with respect to MEMU
0	z4 Instruction Cache	Asynchronous
1	z4 Data Cache	Asynchronous
2	z4 Instruction AHB	Asynchronous
3	z4 Data AHB	Asynchronous
4	Reserved	-
5	Reserved	-
6	Reserved	-
7	Reserved	-
8	DMA AHB	Synchronous
9	HSM AHB	Synchronous
10	Ethernet AHB	Synchronous
11	Flexray AHB	Synchronous
12	Reserved	-
13	Reserved	-
14	Reserved	-
15	Reserved	-
16	PRAM_1 AHB	Asynchronous
17	Reserved	-
18	PRAM_1 DMC	Asynchronous
19	Reserved	-

NOTE

This table shows that the master's AHB could be a source of an uncorrectable RAM error. Thus, if any master AHB from this table accesses a flash-memory location with an uncorrectable ECC error, the MEMU will report both the flash-memory error and a RAM uncorrectable error.

Table 73-6. Corresponding error sources for peripheral RAM OFLW0 register

Bit number	Error source	Clock relationship with respect to MEMU
0	PBRIDGE B	Synchronous
1	DMA TCD	Synchronous

Table continues on the next page...

Table 73-6. Corresponding error sources for peripheral RAM OFLW0 register (continued)

Bit number	Error source	Clock relationship with respect to MEMU
2	HSM instruction cache	Synchronous
3	HSM PRAM	Synchronous
4	PBRIDGE B DMC	Synchronous

Table 73-7. Corresponding error sources for flash memory OFLW0 register

Bit number	Error source	Clock relationship with respect to MEMU
0	PFLASH_0	Asynchronous
1	PFLASH_1	Asynchronous
2	PFLASH_2	Asynchronous

73.1.6 Concurrent overflow register (OFLW) implementation for MEMU_1

Errors are reported to the MEMU from these categories:

- System RAM
- Peripheral RAM

In an overflow condition, it is possible to identify the unique error source that caused the overflow. Corresponding error sources for the OFLW register details the corresponding error source to the respective bit-field in the Concurrent Overflow registers.

Positions not explicitly listed in the tables below are reserved.

Table 73-8. Corresponding error sources for system RAM OFLW0 register

Bit number	Error source	Clock relationship with respect to MEMU
0	z2 Instruction AHB	Synchronous
1	z2 Data AHB	Synchronous
2	PRAM_0 AHB	Asynchronous
3	PRAM_0 DMC	Asynchronous

Table 73-9. Corresponding error sources for peripheral RAM OFLW0 register

Bit number	Error source	Clock relationship with respect to MEMU
0	PBRIDGE A	Synchronous
1	PBRIDGE A DMC	Synchronous

73.1.7 MEMU_0 registers

The following table shows the address offsets and implementation of the MEMU_0 registers in this chip.

Table 73-10. MEMU_0 registers

Offset	Register
0x0000	Control register (CTRL)
0x0004	Error flag register (ERR_FLAG)
0x000C	Debug register (DEBUG)
0x0020	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS0)
0x0024	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR0)
0x0028	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS1)
0x002C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR1)
0x0030	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS2)
0x0034	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR2)
0x0038	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS3)
0x003C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR3)
0x0040	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS4)
0x0044	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR4)
0x0048	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS5)
0x004C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR5)
0x0050	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS6)
0x0054	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR6)
0x0058	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS7)
0x005C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR7)
0x0060	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS8)
0x0064	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR8)
0x0068	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS9)
0x006C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR9)
0x0070	System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)
0x0074	System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)
0x0078	System RAM concurrent overflow register (SYS_RAM_OFLW0)
0x0620	Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS0)

Table continues on the next page...

Table 73-10. MEMU_0 registers (continued)

Offset	Register
0x0624	Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR0)
0x0628	Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS1)
0x062C	Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR1)
0x0630	Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)
0x0634	Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)
0x0638	Peripheral RAM concurrent overflow register (PERIPH_RAM_OFLW0)
0x0C20	Flash memory correctable error reporting table status register (FLASH_CERR_STS0)
0x0C24	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR0)
0x0C28	Flash memory correctable error reporting table status register (FLASH_CERR_STS1)
0x0C2C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR1)
0x0C30	Flash memory correctable error reporting table status register (FLASH_CERR_STS2)
0x0C34	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR2)
0x0C38	Flash memory correctable error reporting table status register (FLASH_CERR_STS3)
0x0C3C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR3)
0x0C40	Flash memory correctable error reporting table status register (FLASH_CERR_STS4)
0x0C44	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR4)
0x0C48	Flash memory correctable error reporting table status register (FLASH_CERR_STS5)
0x0C4C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR5)
0x0C50	Flash memory correctable error reporting table status register (FLASH_CERR_STS6)
0x0C54	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR6)
0x0C58	Flash memory correctable error reporting table status register (FLASH_CERR_STS7)
0x0C5C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR7)
0x0C60	Flash memory correctable error reporting table status register (FLASH_CERR_STS8)
0x0C64	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR8)
0x0C68	Flash memory correctable error reporting table status register (FLASH_CERR_STS9)
0x0C6C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR9)
0x0C70	Flash memory correctable error reporting table status register (FLASH_CERR_STS10)
0x0C74	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR10)
0x0C78	Flash memory correctable error reporting table status register (FLASH_CERR_STS11)
0x0C7C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR11)
0x0C80	Flash memory correctable error reporting table status register (FLASH_CERR_STS12)
0x0C84	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR12)
0x0C88	Flash memory correctable error reporting table status register (FLASH_CERR_STS13)
0x0C8C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR13)
0x0C90	Flash memory correctable error reporting table status register (FLASH_CERR_STS14)
0x0C94	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR14)
0x0C98	Flash memory correctable error reporting table status register (FLASH_CERR_STS15)
0x0C9C	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR15)
0x0CA0	Flash memory correctable error reporting table status register (FLASH_CERR_STS16)

Table continues on the next page...

Table 73-10. MEMU_0 registers (continued)

Offset	Register
0x0CA4	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR16)
0x0CA8	Flash memory correctable error reporting table status register (FLASH_CERR_STS17)
0x0CAC	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR17)
0x0CB0	Flash memory correctable error reporting table status register (FLASH_CERR_STS18)
0x0CB4	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR18)
0x0CB8	Flash memory correctable error reporting table status register (FLASH_CERR_STS19)
0x0CBC	Flash memory correctable error reporting table address register (FLASH_CERR_ADDR19)
0x0CC0	Flash memory uncorrectable error reporting table status register (FLASH_UNCERR_STS)
0x0CC4	Flash memory uncorrectable error reporting table address register (FLASH_UNCERR_ADDR)
0x0CC8	Flash memory concurrent overflow register (FLASH_OFLW0)

73.1.8 MEMU_1 registers

The following table shows the address offsets and implementation of the MEMU_1 registers in this chip.

Table 73-11. MEMU_1 registers

Offset	Register
0x0000	Control register (CTRL)
0x0004	Error flag register (ERR_FLAG)
0x000C	Debug register (DEBUG)
0x0020	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS0)
0x0024	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR0)
0x0028	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS1)
0x002C	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR1)
0x0030	System RAM correctable error reporting table status register (SYS_RAM_CERR_STS2)
0x0034	System RAM correctable error reporting table address register (SYS_RAM_CERR_ADDR2)
0x0038	System RAM uncorrectable error reporting table status register (SYS_RAM_UNCERR_STS)
0x003C	System RAM uncorrectable error reporting table address register (SYS_RAM_UNCERR_ADDR)
0x0040	System RAM concurrent overflow register (SYS_RAM_OFLW0)
0x0620	Peripheral RAM correctable error reporting table status register (PERIPH_RAM_CERR_STS0)
0x0624	Peripheral RAM correctable error reporting table address register (PERIPH_RAM_CERR_ADDR0)
0x0628	Peripheral RAM uncorrectable error reporting table status register (PERIPH_RAM_UNCERR_STS)
0x062C	Peripheral RAM uncorrectable error reporting table address register (PERIPH_RAM_UNCERR_ADDR)
0x0630	Peripheral RAM concurrent overflow register (PERIPH_RAM_OFLW0)

NOTE

Reserved locations with offset values as 0xC20, 0xC24, 0xC28, 0xC2C, and 0xC30 would not give error on access.

73.2 Introduction

The MEMU is responsible for collection and reporting of error events associated with ECC (Error Correction Code) logic used on:

- System RAM
- Peripheral RAM
- Flash memory

When any of the following events occur the MEMU receives an error signal which causes an event to be recorded and corresponding error flags to be set and reported to FCCU (Fault Collection and Control Unit).

- Correctable Error: It comprises the following:
  - Single Bit error in the data part that is detected via ECC for:
    - System RAM
    - Peripheral RAM
    - Flash memory
  - Single Bit error in the data part that is detected via MBIST on any RAM.
- Uncorrectable Error: It comprises the following:
  - Multiple bit error that is detected via ECC for:
    - System RAM
    - Peripheral RAM
    - Flash memory
  - Multiple bit error that is detected via MBIST on any SRAM. Based on MBIST implementation, the MBIST might report several correctable errors in the same word instead of an uncorrectable one. It is the task of the software after MBIST to aggregate these reports and detect the uncorrectable error.
  - Addressing errors and unused data bit errors detected by ECC logic.

The MEMU system connections are chip-specific; see the chip-specific MEMU information.

When multiple errors are indicated from various sources at same instant, an Overflow can be indicated by the MEMU to the FCCU. Overflow can also be indicated if the reporting table entries are full and a new unique error is reported by the system.

MEMU processes the errors based on whether they are correctable or uncorrectable (from either ECC or MBIST logic), independent of the source generating these errors.

The MEMU has multiple instances of the basic reporting block; one each for:

- System RAM ECC and MBIST
- Peripheral RAM ECC
- Flash memory ECC

Each instance has two reporting tables. Correctable errors are reported in one table, and the uncorrectable errors are reported in the other.

See Design overview for the reporting block details.

73.3 Features

The MEMU has the following features:

- Supports up to 128 error sources per the following reporting blocks (the number of error sources are chip-specific; see the chip-specific MEMU information):
  - System RAM ECC and MBIST
  - Peripheral RAM ECC
  - Flash memory ECC
- Support for error reporting from the following category of error sources (both for ECC and MBIST):
  - System RAM
  - Peripheral RAM
  - Flash memory
- Unique reported errors are logged into the module's reporting table which is accessible to CPU via memory mapped CPU register programming interface.
- MEMU handles overflow during error assertion and reports status accordingly.
- Unique errors are stored in correctable and non correctable section of the reporting table and corresponding indication is generated to the FCCU.
- CPU can program the known errors into the reporting table to avoid their re-reporting by MEMU.
- CPU can clear the reporting table errors.
- The reporting table for uncorrectable errors supports only 1 entry.

73.4 Block diagram

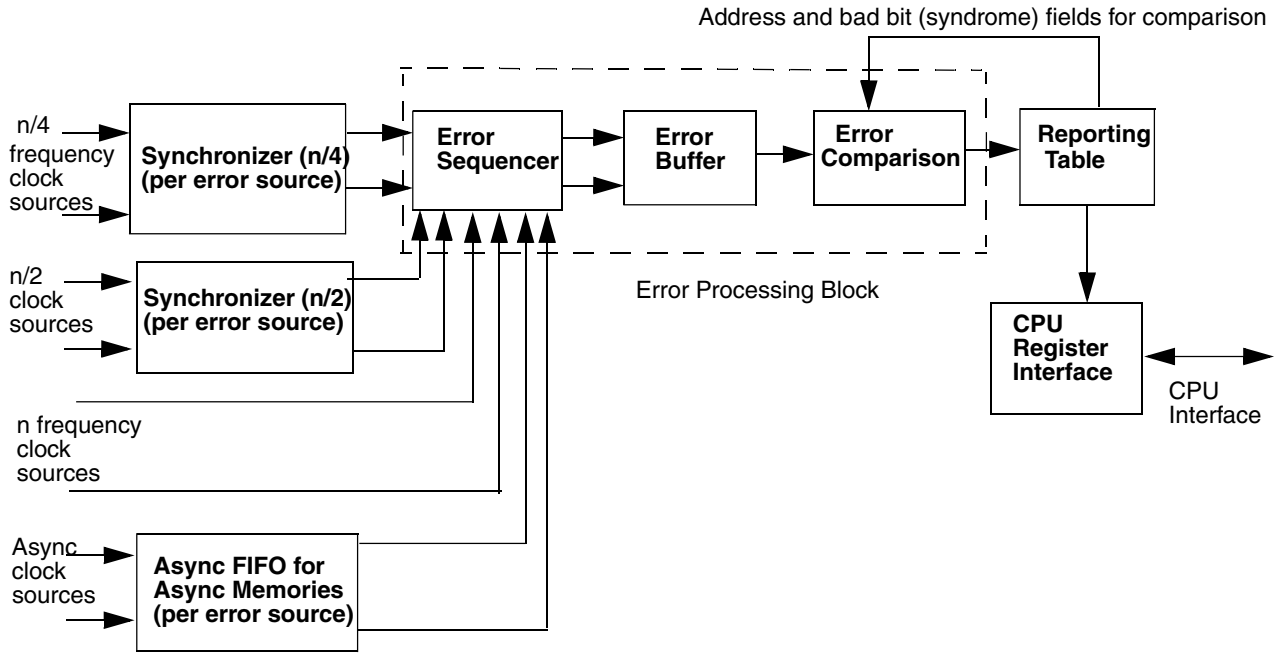


Figure 73-3. MEMU block diagram

73.5 Design overview

The MEMU should be enabled on power-on reset. The user software can write known error addresses into the reporting table to prevent reporting of those errors addresses to FCCU should they later be accessed in operation. All writes into the reporting table by the CPU that assert the Valid Bit (of the respective reporting table entry) will be indicated as a new error detected and will be reported to the FCCU. This feature can also be used as self-checking option of the MEMU and the MEMU-FCCU connection.

The MEMU design does not limit any particular sequence or errors. See Handling overflows (Multiple error reporting), for details on overflow behaviors.

The basic algorithm flow/architecture for error reporting in the MEMU is summarized below:

1. Synchronizer block

   For asynchronous inputs, a FIFO-based structure (per asynchronous channel) is implemented. All the error inputs (correctable error reported, uncorrectable error reported, error address and bad bits/syndrome) are concatenated and stored in the

FIFO. While storing the entry in the FIFO, a duplicity check is performed with the previously stored entry (the entry which was stored in the last cycle). If both the entries are duplicate the new entry is deleted and the pointers are rolled back to the old value, otherwise the new entry is stored. Whenever there is an overflow (FULL condition) in the Asynchronous FIFO, it is denoted by setting the Error Buffer Overflow or the EBO flag to the FCCU and the corresponding bit in the Concurrent Overflow Register is set.

For synchronous inputs, inputs are sampled whenever the corresponding clock synchronizing signal is asserted for a particular channel.

2. Error Processing block

This block is used to process the incoming errors coming from the device.

- Error Sequencer is used for sequencing the errors so that these are processed one per clock. In case of more than one errors detected by the sequencer logic, one is sent for the processing (error comparison block), the other is stored in the error buffer and the rest are marked as overflows in the Concurrent overflow register.
- All the sequences of incoming errors are supported. However, in case of multiple errors being asserted in back to back clock cycles, MEMU will process the errors being forwarded by the sequencer logic and register the rest of the errors as Concurrent Overflows in the Concurrent Overflow Register.

  For example, the MEMU processes a 0-2-2 sequence like this:

  1. The MEMU sends the first error directly to the processing.
  2. The MEMU stores the second error in the error buffer.
  3. The MEMU performs a uniqueness check, comparing the third error with the existing entry in the error buffer for uniqueness.

  If the errors are unique, the MEMU:

  - Classifies the fourth error as an overflow
  - Sets the corresponding EBO flag in Error flag register (MEMU_ERR_FLAG)

  If the errors are not unique, the MEMU:

  - Drops the third error
  - Stores the fourth error in the error buffer
  - Does not set any EBO flag in Error flag register (MEMU_ERR_FLAG)

- While storing the error in the error buffer, uniqueness is determined by comparing the error address, correctable or uncorrectable error type and the bad bit/syndrome (depending on ECC or MBIST logic) of the incoming error with

the one stored in buffer. If the error buffer is full, and the incoming error to be stored in the error buffer is not yet in the error table, Error Buffer Overflow (EBO) is registered and the corresponding bit in the Concurrent Overflow Register is set., else the incoming error is dropped.

- While doing the error buffer uniqueness check, the type of error i.e Correctable Error (CE) or Uncorrectable Error UCE) is also compared, hence even if the Error Address and the Bad bit/syndrome fields (Depending on whether the incoming error is generated from ECC or MBIST logic) of the incoming error matches with the error buffer, if the type of errors (CE and UCE) differ they are treated as unique
- Determine the type and uniqueness of errors in the error comparison block and store them in the correct reporting table (correctable or uncorrectable)
- Generate signals to assert the necessary flags. Be aware that it takes several MEMU clock cycles (<=5 unless the error gets temporarily stored in the error buffer due to a collision) for an error report to cause a signal to be sent to the FCCU.

3. Reporting table

   This is a pre-defined table in the MEMU to store the details for the device. For the reporting table if the entry is unique it is stored in the reporting table. If the entry is not unique it is not stored and discarded.

   The MEMU asserts proper flags when error is stored in the reporting table or an overflow is asserted. Flags are also asserted when the software writes the VLD bit to the reporting table.

   The CPU can clear the flags signaling errors or overflows to the FCCU directly by writing 1 at the corresponding valid bit of the status register (i.e. ..._CERR_STS or ..._UNCERR_STS). The clearing of flags is totally independent from the status of VLD bits in the reporting table.

   The reporting table sizes are chip-specific; see the chip-specific MEMU information.

4. Register block

   The CPU register programming interface provides the memory-mapped registers necessary for the CPU to access the reporting table and other control and status registers for the modules. Flags to FCCU will be asserted on write by CPU to the valid bit or by MEMU itself if a unique error is stored. However, in the situation when both MEMU and the CPU are accessing the same Register location, a wait cycle will be signaled to the CPU and the CPU command will be completed in the next clock cycle.

The software, in principle, is not supposed to write VLD to 1 if it is already 1.

The CPU register interface provides the decoding of the peripheral signals to allow access to the reporting table and other registers in module.

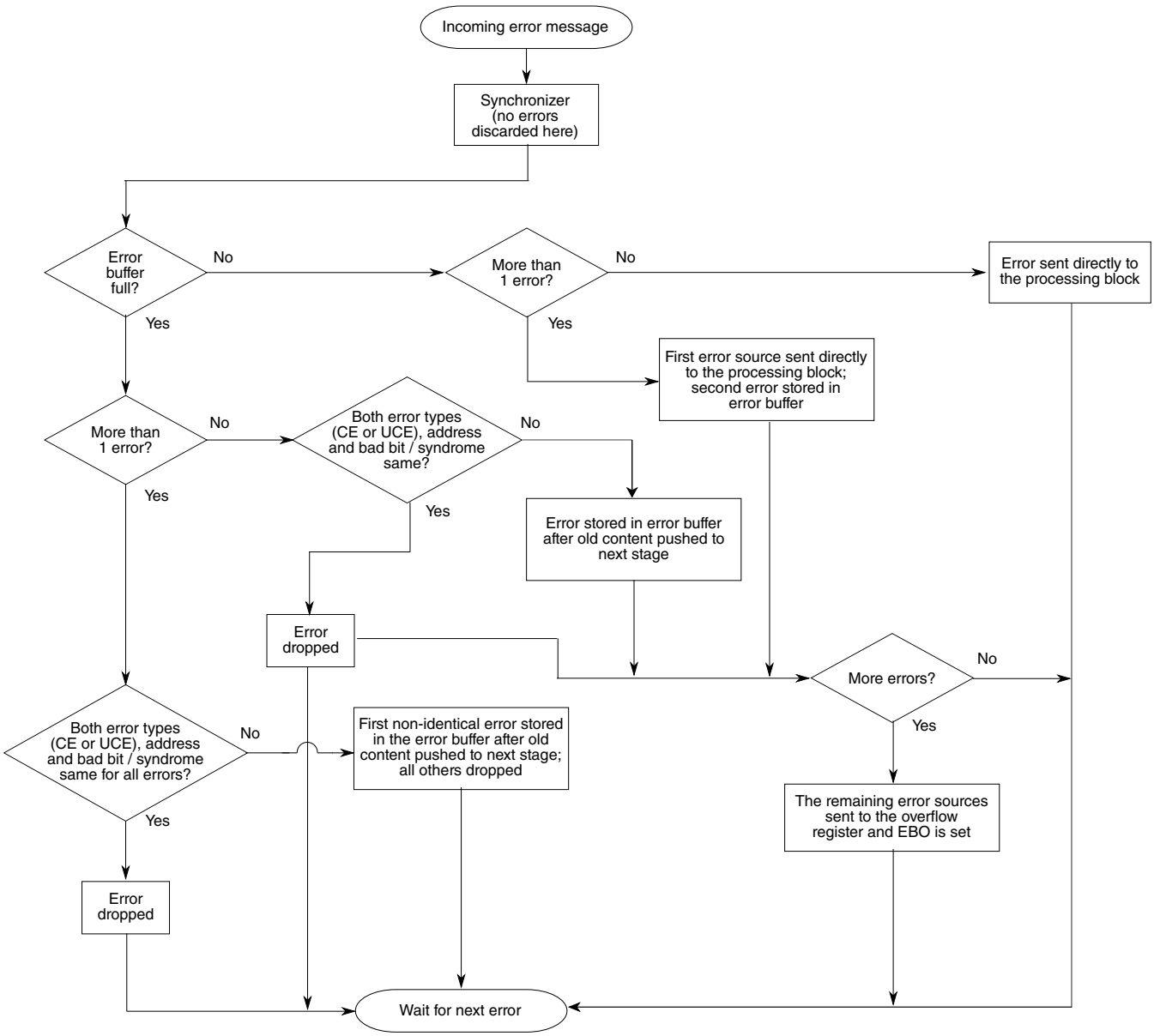


Figure 73-4. MEMU error buffer uniqueness check flow chart

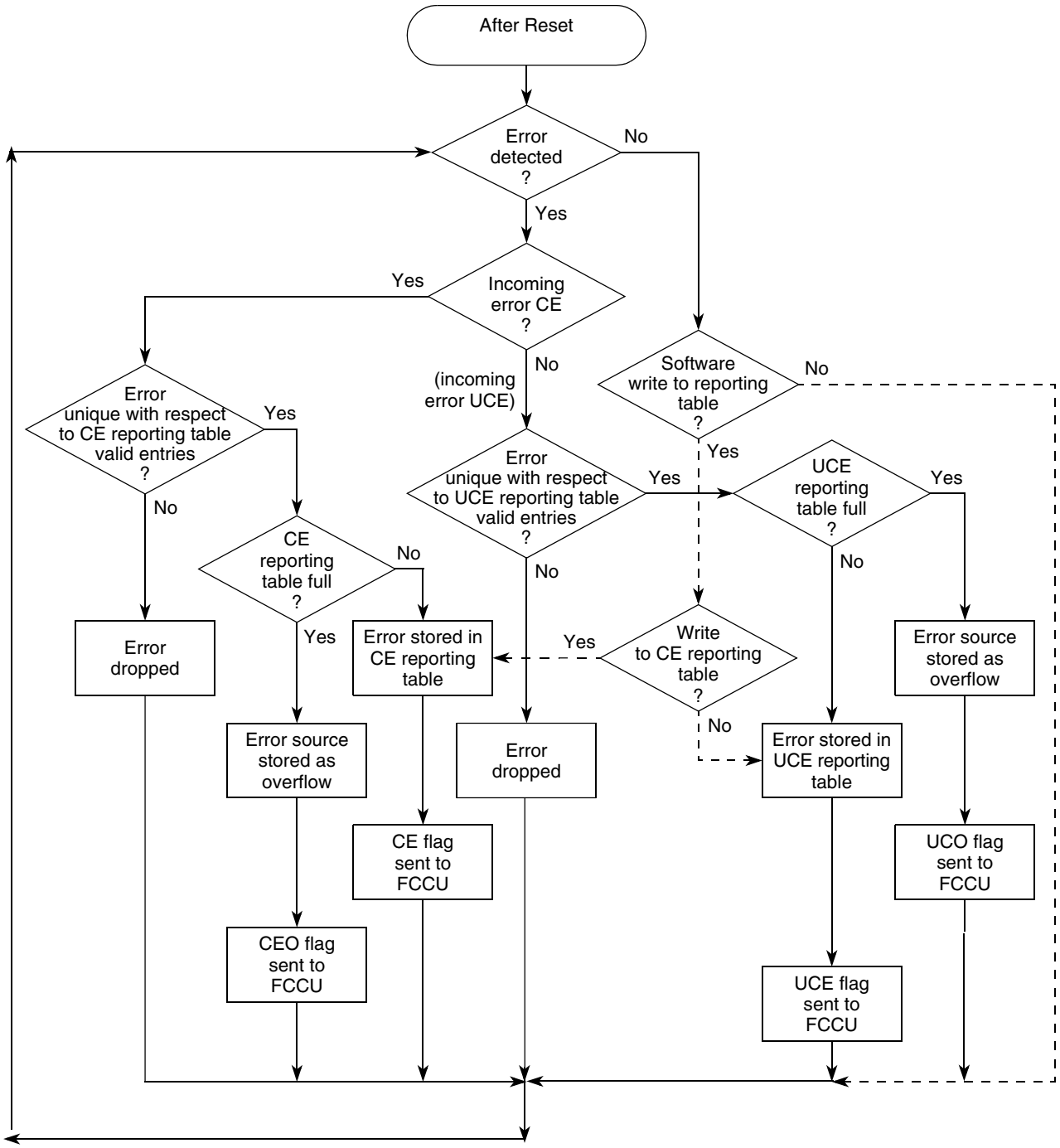


Figure 73-5. MEMU reporting table uniqueness check flow chart

73.6 External signal description

MEMU has no external signals.

73.7 Memory map and register definition

This section describes the registers on this module.

73.7.1 Overview

The memory map comprises 32-bit aligned registers that can be accessed via 8-bit,16-bit, or 32-bit accesses. Write access to reserved address locations will generate a transfer error. Read data from reserved locations will also generate a transfer error and the read data bus will return all 0's.

NOTE

The module does not check for correctness of the programmed values in registers. Software must ensure that the correct values are being written.

NOTE

The memory map in this section shows all the possible registers on this module and the associated offsets (which vary depending on the number of registers actually implemented). The actual availability, number and addresses of registers, and number of reported errors in the reporting table are chip-specific; see the Device Configuration chapter that describes how modules are configured and connected.

MEMU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control register (MEMU_CTRL)	32	R/W	0000_0000h	73.7.2/3106
4	Error flag register (MEMU_ERR_FLAG)	32	w1c	0000_0000h	73.7.3/3106
C	Debug register (MEMU_DEBUG)	32	R/W	0000_0000h	73.7.4/3110
20	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS0)	32	R/W	0000_0000h	73.7.5/3112
24	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR0)	32	R/W	0000_0000h	73.7.6/3113
28	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS1)	32	R/W	0000_0000h	73.7.5/3112
2C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR1)	32	R/W	0000_0000h	73.7.6/3113
30	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS2)	32	R/W	0000_0000h	73.7.5/3112

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
34	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR2)	32	R/W	0000_0000h	73.7.6/3113
38	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS3)	32	R/W	0000_0000h	73.7.5/3112
3C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR3)	32	R/W	0000_0000h	73.7.6/3113
40	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS4)	32	R/W	0000_0000h	73.7.5/3112
44	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR4)	32	R/W	0000_0000h	73.7.6/3113
48	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS5)	32	R/W	0000_0000h	73.7.5/3112
4C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR5)	32	R/W	0000_0000h	73.7.6/3113
50	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS6)	32	R/W	0000_0000h	73.7.5/3112
54	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR6)	32	R/W	0000_0000h	73.7.6/3113
58	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS7)	32	R/W	0000_0000h	73.7.5/3112
5C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR7)	32	R/W	0000_0000h	73.7.6/3113
60	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS8)	32	R/W	0000_0000h	73.7.5/3112
64	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR8)	32	R/W	0000_0000h	73.7.6/3113
68	System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS9)	32	R/W	0000_0000h	73.7.5/3112
6C	System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR9)	32	R/W	0000_0000h	73.7.6/3113
70	System RAM uncorrectable error reporting table status register (MEMU_SYS_RAM_UNCERR_STS)	32	R/W	0000_0000h	73.7.7/3113
74	System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR)	32	R/W	0000_0000h	73.7.8/3114
78	System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW0)	32	w1c	0000_0000h	73.7.9/3114
620	Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS0)	32	R/W	0000_0000h	73.7.10/ 3115
624	Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR0)	32	R/W	0000_0000h	73.7.11/ 3116
628	Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS1)	32	R/W	0000_0000h	73.7.10/ 3115
62C	Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR1)	32	R/W	0000_0000h	73.7.11/ 3116

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
630	Peripheral RAM uncorrectable error reporting table status register (MEMU_PERIPH_RAM_UNCERR_STS)	32	R/W	0000_0000h	73.7.12/ 3116
634	Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR)	32	R/W	0000_0000h	73.7.13/ 3117
638	Peripheral RAM concurrent overflow register (MEMU_PERIPH_RAM_OFLW0)	32	w1c	0000_0000h	73.7.14/ 3117
C20	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS0)	32	R/W	0000_0000h	73.7.15/ 3118
C24	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR0)	32	R/W	0000_0000h	73.7.16/ 3119
C28	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS1)	32	R/W	0000_0000h	73.7.15/ 3118
C2C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR1)	32	R/W	0000_0000h	73.7.16/ 3119
C30	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS2)	32	R/W	0000_0000h	73.7.15/ 3118
C34	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR2)	32	R/W	0000_0000h	73.7.16/ 3119
C38	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS3)	32	R/W	0000_0000h	73.7.15/ 3118
C3C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR3)	32	R/W	0000_0000h	73.7.16/ 3119
C40	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS4)	32	R/W	0000_0000h	73.7.15/ 3118
C44	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR4)	32	R/W	0000_0000h	73.7.16/ 3119
C48	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS5)	32	R/W	0000_0000h	73.7.15/ 3118
C4C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR5)	32	R/W	0000_0000h	73.7.16/ 3119
C50	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS6)	32	R/W	0000_0000h	73.7.15/ 3118
C54	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR6)	32	R/W	0000_0000h	73.7.16/ 3119
C58	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS7)	32	R/W	0000_0000h	73.7.15/ 3118
C5C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR7)	32	R/W	0000_0000h	73.7.16/ 3119
C60	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS8)	32	R/W	0000_0000h	73.7.15/ 3118
C64	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR8)	32	R/W	0000_0000h	73.7.16/ 3119
C68	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS9)	32	R/W	0000_0000h	73.7.15/ 3118

Table continues on the next page...

MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C6C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR9)	32	R/W	0000_0000h	73.7.16/ 3119
C70	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS10)	32	R/W	0000_0000h	73.7.15/ 3118
C74	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR10)	32	R/W	0000_0000h	73.7.16/ 3119
C78	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS11)	32	R/W	0000_0000h	73.7.15/ 3118
C7C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR11)	32	R/W	0000_0000h	73.7.16/ 3119
C80	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS12)	32	R/W	0000_0000h	73.7.15/ 3118
C84	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR12)	32	R/W	0000_0000h	73.7.16/ 3119
C88	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS13)	32	R/W	0000_0000h	73.7.15/ 3118
C8C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR13)	32	R/W	0000_0000h	73.7.16/ 3119
C90	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS14)	32	R/W	0000_0000h	73.7.15/ 3118
C94	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR14)	32	R/W	0000_0000h	73.7.16/ 3119
C98	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS15)	32	R/W	0000_0000h	73.7.15/ 3118
C9C	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR15)	32	R/W	0000_0000h	73.7.16/ 3119
CA0	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS16)	32	R/W	0000_0000h	73.7.15/ 3118
CA4	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR16)	32	R/W	0000_0000h	73.7.16/ 3119
CA8	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS17)	32	R/W	0000_0000h	73.7.15/ 3118
CAC	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR17)	32	R/W	0000_0000h	73.7.16/ 3119
CB0	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS18)	32	R/W	0000_0000h	73.7.15/ 3118
CB4	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR18)	32	R/W	0000_0000h	73.7.16/ 3119
CB8	Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS19)	32	R/W	0000_0000h	73.7.15/ 3118
CBC	Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR19)	32	R/W	0000_0000h	73.7.16/ 3119
CC0	Flash memory uncorrectable error reporting table status register (MEMU_FLASH_UNCERR_STS)	32	R/W	0000_0000h	73.7.17/ 3119

Table continues on the next page...

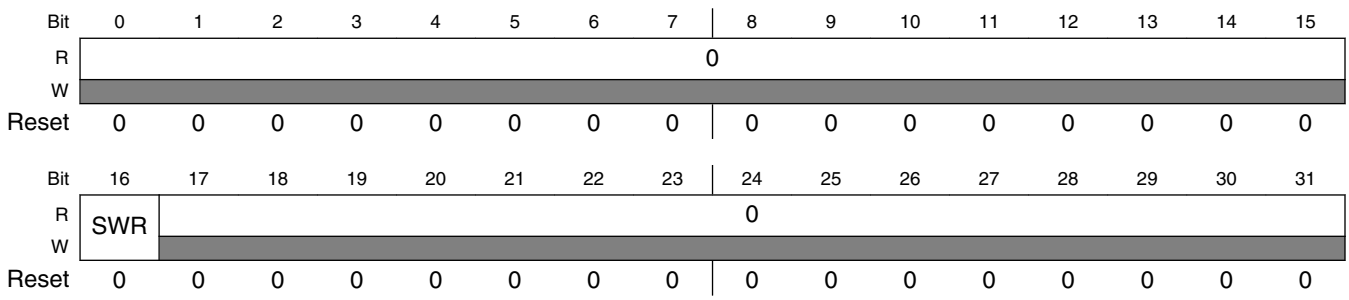
MEMU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
CC4	Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR)	32	R/W	0000_0000h	73.7.18/ 3120
CC8	Flash memory concurrent overflow register (MEMU_FLASH_OFLW0)	32	w1c	0000_0000h	73.7.19/ 3120

73.7.2 Control register (MEMU_CTRL)

Access: User read/write

Address: 0h base + 0h offset = 0h



MEMU_CTRL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 SWR	Software Reset bit. Setting this will clear status flags and overflow register. However, the reporting table registers are not cleared. This bit will auto clear on next clock cycle. 0 No reset. 1 Reset asserted.
17–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

73.7.3 Error flag register (MEMU_ERR_FLAG)

Bits in the ERR_FLAG register indicate:

- A new entry in an error reporting table has been made (indicated by PR_CE, PR_UCE, F_CE, F_UCE, SR_CE, and SR_UCE)

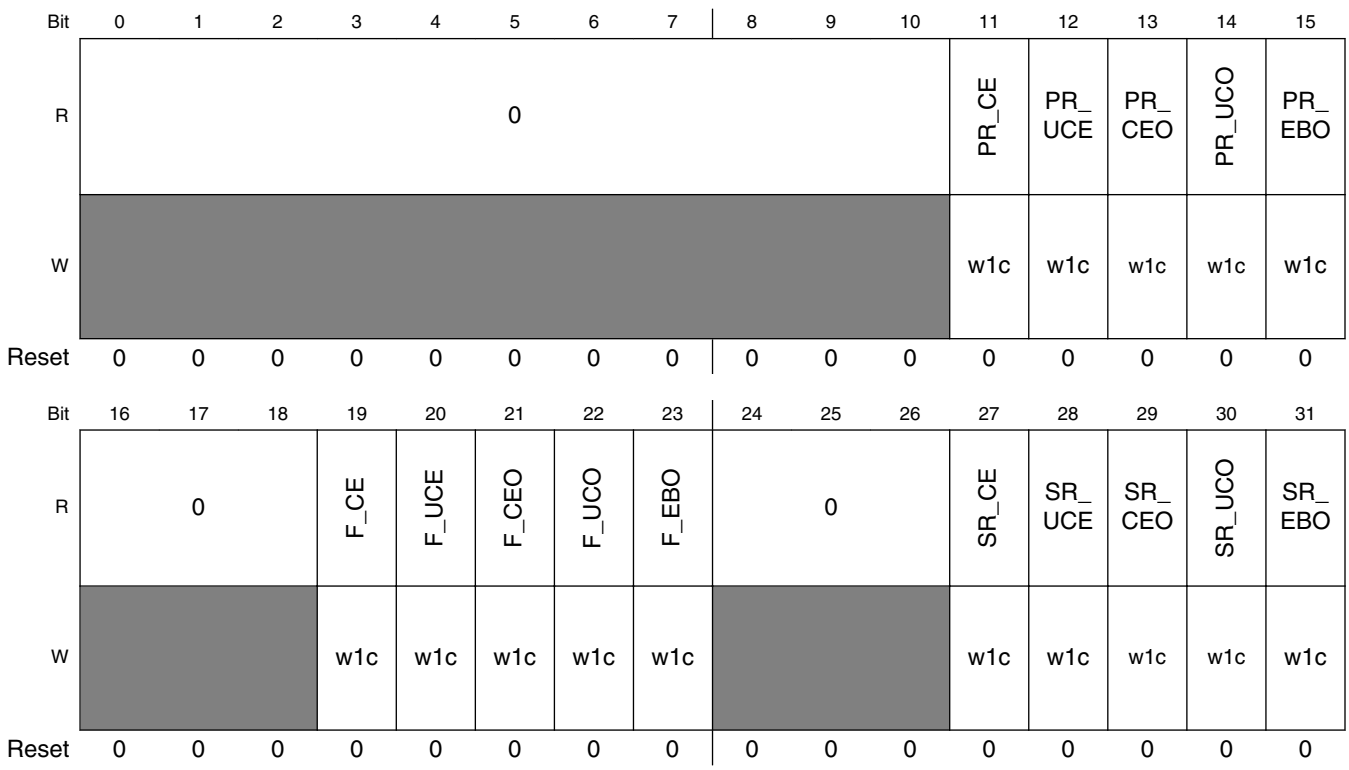
- An overflow condition has been encountered when attempting to make a new entry in an error reporting table (indicated by SR_UCO, SR_CEO, F_UCO,F_CEO, PR_UCO, and PR_CEO)
- An overflow condition has been encountered in the internal error processing buffer of a MEMU reporting block.

Individual flags can be cleared by writing a '1'. This also resets the corresponding error indication to FCCU. Flags will not automatically reset if entries are removed from the reporting table. They have to be explicitly cleared by SW.

Bits SR_UCO, SR_CEO, F_UCO, F_CEO, PR_UCO, and PR_CEO are asserted when an overflow in the uncorrectable error reporting table is detected.

Access: User read/write

Address: 0h base + 4h offset = 4h



MEMU_ERR_FLAG field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 PR_CE	Peripheral RAM ECC Correctable Error detect flag. Indicates that a new and unique Peripheral RAM ECC correctable error is detected. Asserted when a new entry is inserted into the correctable error reporting table (either by CPU or module).

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
	0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
12 PR_UCE	Peripheral RAM ECC Uncorrectable Error Detect flag. Asserted when a new Peripheral RAM ECC uncorrectable error was detected or a new entry is inserted into the uncorrectable error reporting table (either by CPU or module). 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
13 PR_CEO	Peripheral RAM ECC Correctable error Overflow flag. Asserted when the Peripheral RAM ECC correctable error reporting table is full and a new entry is attempted to be made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
14 PR_UCO	Peripheral RAM ECC Uncorrectable error Overflow flag. Asserted when the Peripheral RAM ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
15 PR_EBO	Peripheral RAM ECC Error buffer Overflow flag. Asserted when the internal error processing buffer of Peripheral RAM ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 F_CE	Flash ECC Correctable Error Detect flag. Indicates that a new and unique Flash ECC correctable error was detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
20 F_UCE	Flash ECC Uncorrectable Error Detect flag. Asserted when: • a new Flash ECC uncorrectable error was detected. • a new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
21 F_CEO	Flash ECC Correctable Error Overflow flag. Asserted when the Flash ECC correctable error reporting table is full and a new entry is made to this table.

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
	0 No Overflow. 1 Overflow in the correctable error reporting table detected.
22 F_UCO	Flash ECC Uncorrectable Error Overflow flag. Asserted when the Flash ECC uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
23 F_EBO	Flash ECC Error buffer Overflow. Flag Asserted when the internal error processing buffer of Flash ECC MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for MEMU to process them all. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 SR_CE	System RAM ECCand MBIST Correctable Error detect flag. Indicates that a new and unique System RAM ECCand MBISTcorrectable error is detected. Asserted when a new entry to the correctable error reporting table is done (either by CPU or module). 0 No new and unique error detected. 1 New entry in correctable error reporting table is made.
28 SR_UCE	System RAM ECCand MBIST Uncorrectable Error Detect flag. Asserted when: • A new System RAM ECCand MBIST uncorrectable error is detected • A new entry to the uncorrectable error reporting table is done (either by CPU or module) 0 No new and unique error detected. 1 New entry in uncorrectable error reporting table is made.
29 SR_CEO	System RAM ECCand MBIST Correctable error Overflow flag. Asserted when the System RAM ECCand MBIST correctable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the correctable error reporting table detected.
30 SR_UCO	System RAM ECCand MBIST Uncorrectable error Overflow flag. Asserted when the System RAM ECCand MBIST uncorrectable error reporting table is full and a new entry is made to this table. 0 No Overflow. 1 Overflow in the uncorrectable error reporting table detected.
31 SR_EBO	System RAM ECCand MBIST Error buffer Overflow.

Table continues on the next page...

MEMU_ERR_FLAG field descriptions (continued)

Field	Description
	Flag asserted when the internal error processing buffer of System RAM ECCand MBIST MEMU reporting block has overflowed. This can happen when too many errors are reported in a short interval of time for processing by MEMU. The channel(s) from which error reports were dropped due to the overflow are indicated in the XXXXX_OFLW registers. This field will also be set to 1 if the input ASYNC FIFOs overflow. 0 No Overflow. 1 Overflow in the internal error processing buffer detected.

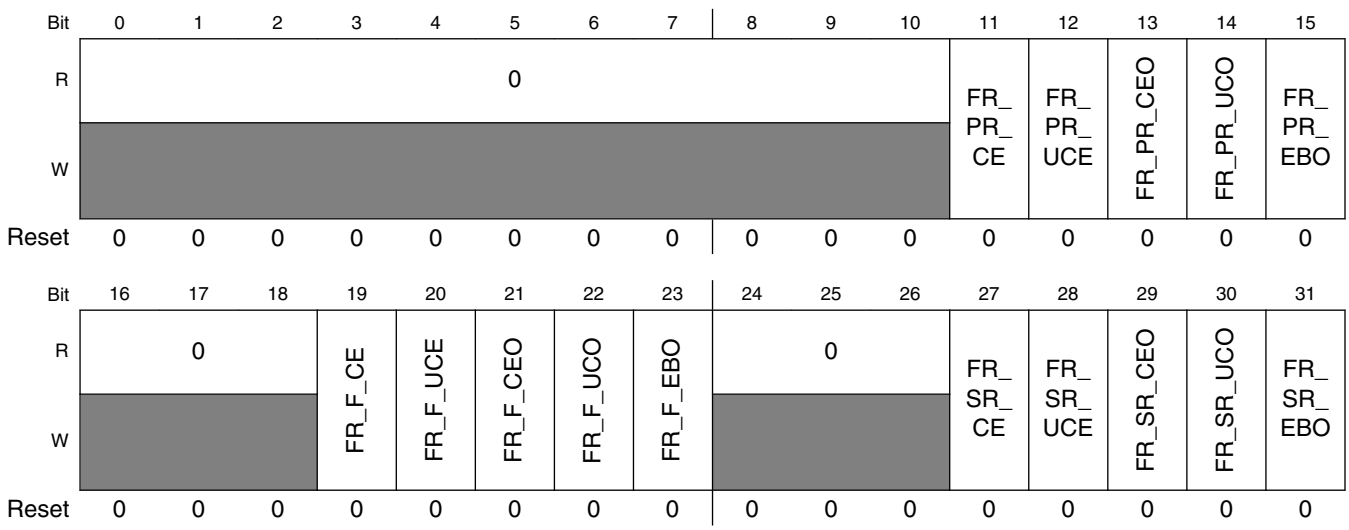
73.7.4 Debug register (MEMU_DEBUG)

This register is used to check the connections between MEMU & FCCU.

The error output of the MEMU towards FCCU shall stay set if forced until reset by SW via the MEMU in the normal way.

Access: User read/write

Address: 0h base + Ch offset = Ch



MEMU_DEBUG field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 FR_PR_CE	Force Peripheral RAM Correctable Error detect flag. 0 Forcing is disabled. 1 Forces PR_CE flag going towards FCCU to 1.

Table continues on the next page...

MEMU_DEBUG field descriptions (continued)

Field	Description
12 FR_PR_UCE	Force Peripheral RAM Uncorrectable Error detect flag. 0 Forcing is disabled. 1 Forces PR_UCE flag going towards FCCU to 1.
13 FR_PR_CEO	Force Peripheral RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces PR_CEO flag going towards FCCU to 1.
14 FR_PR_UCO	Forces Peripheral RAM Uncorrectable Error overflow flag. 0 Forcing is disabled. 1 Forces PR_UCO flag going towards FCCU to 1.
15 FR_PR_EBO	Forces Peripheral RAM Error Buffer Overflow Flag. 0 Forcing is disabled. 1 Forces PR_EBO flag going towards FCCU to 1.
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 FR_F_CE	Forces Flash Correctable Error detect flag 0 Forcing is disabled. 1 Forces F_CE flag going towards FCCU to 1.
20 FR_F_UCE	Forces Flash Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces F_UCE flag going towards FCCU to 1.
21 FR_F_CEO	Forces Flash Correctable Error overflow flag 0 Forcing is disabled. 1 Forces F_CEO flag going towards FCCU to 1.
22 FR_F_UCO	Forces Flash Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces F_UCO flag going towards FCCU to 1.
23 FR_F_EBO	Forces Flash Error buffer Overflow flag 0 Forcing is disabled. 1 Forces F_EBO flag going towards FCCU to 1.
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 FR_SR_CE	Forces System RAM Correctable Error detect flag 0 Forcing is disabled. 1 Forces SR_CE flag going towards FCCU to 1.
28 FR_SR_UCE	Forces System RAM Uncorrectable Error detect flag 0 Forcing is disabled. 1 Forces SR_UCE flag going towards FCCU to 1.

Table continues on the next page...

MEMU_DEBUG field descriptions (continued)

Field	Description
29 FR_SR_CEO	Forces System RAM Correctable Error overflow flag 0 Forcing is disabled. 1 Forces SR_CEO flag going towards FCCU to 1.
30 FR_SR_UCO	Forces System RAM Uncorrectable Error overflow flag 0 Forcing is disabled. 1 Forces SR_UCO flag going towards FCCU to 1.
31 FR_SR_EBO	Forces System RAM Error buffer Overflow Flag 0 Forcing is disabled. 1 Forces SR_EBO flag going towards FCCU to 1.

73.7.5 System RAM correctable error reporting table status register (MEMU_SYS_RAM_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 20h offset + (8d × i), where i=0d to 9d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF.

Table continues on the next page...

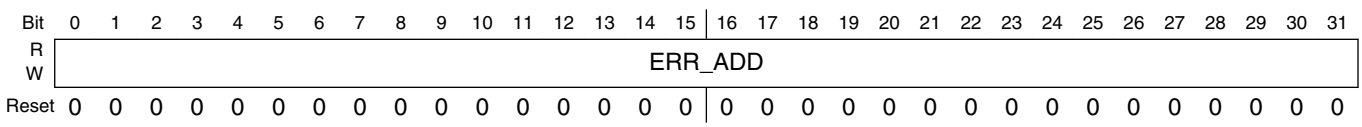
MEMU_SYS_RAM_CERR_STS*n* field descriptions (continued)

Field	Description
	Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

73.7.6 System RAM correctable error reporting table address register (MEMU_SYS_RAM_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + 24h offset + (8d × i), where i=0d to 9d



MEMU_SYS_RAM_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

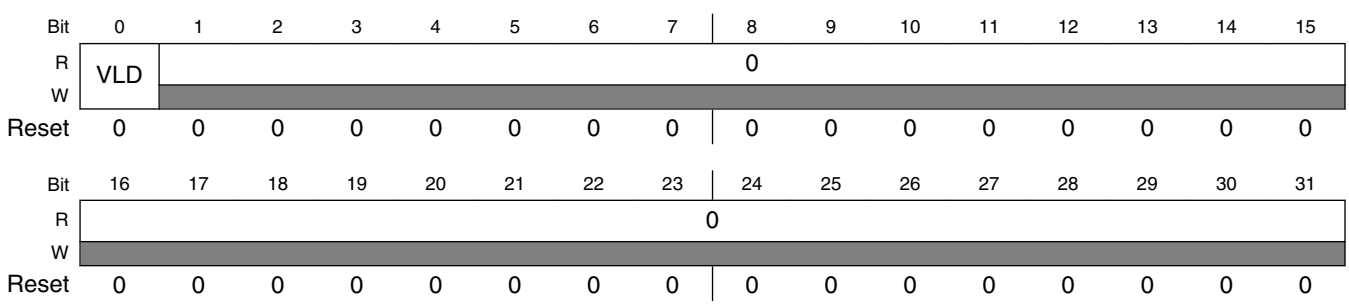
73.7.7 System RAM uncorrectable error reporting table status register (MEMU_SYS_RAM_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 70h offset = 70h



MEMU_SYS_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

73.7.8 System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected.

73.7.9 System RAM concurrent overflow register (MEMU_SYS_RAM_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 78h offset + (4d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_SYS_RAM_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

73.7.10 Peripheral RAM correctable error reporting table status register (MEMU_PERIPH_RAM_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 620h offset + (8d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

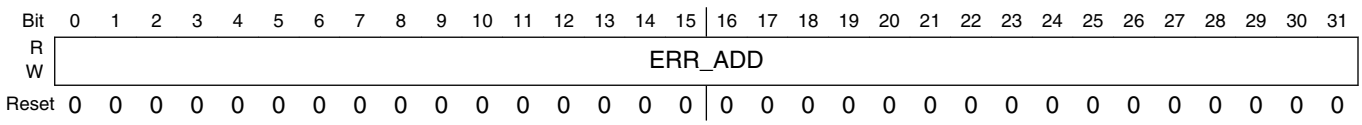
MEMU_PERIPH_RAM_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in RAM where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

73.7.11 Peripheral RAM correctable error reporting table address register (MEMU_PERIPH_RAM_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + 624h offset + (8d × i), where i=0d to 1d



MEMU_PERIPH_RAM_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

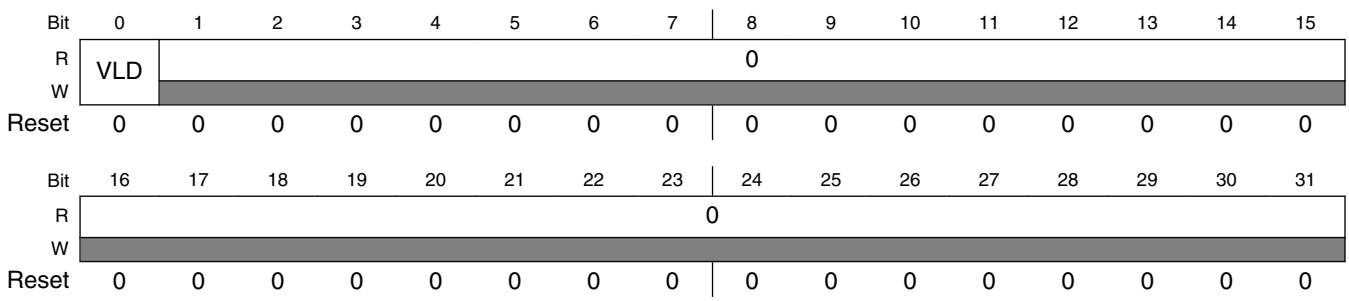
73.7.12 Peripheral RAM uncorrectable error reporting table status register (MEMU_PERIPH_RAM_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 630h offset = 630h



MEMU_PERIPH_RAM_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted.

Table continues on the next page...

MEMU_PERIPH_RAM_UNCERR_STS field descriptions (continued)

Field	Description
	0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

73.7.13 Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 634h offset = 634h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_PERIPH_RAM_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected.

73.7.14 Peripheral RAM concurrent overflow register (MEMU_PERIPH_RAM_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + 638h offset + (4d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_PERIPH_RAM_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

73.7.15 Flash memory correctable error reporting table status register (MEMU_FLASH_CERR_STS*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + C20h offset + (8d × i), where i=0d to 19d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								BAD_BIT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

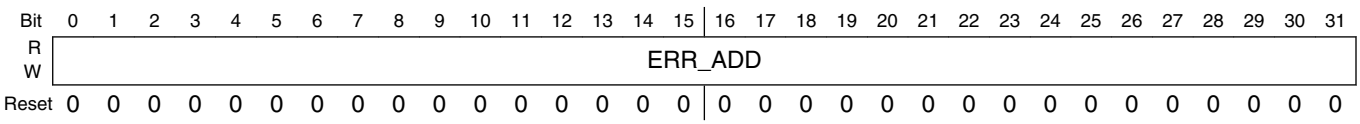
MEMU_FLASH_CERR_STS*n* field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid . Assertion of this bit causes the correctable error detect flag to be asserted. 0 Entry in table is invalid. 1 Entry in table is valid.
1–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 BAD_BIT	Bad bit field. Indicates the bit position in flash memory where the error is detected. This is also known as the syndrome. For non-BIST type of errors, this field is not used and reads as 0xFF. Any value other than 0xFF - Position of the bit in RAM where error is found. 0xFF - Invalid Bad Bit. This field should not be used when processing errors.

73.7.16 Flash memory correctable error reporting table address register (MEMU_FLASH_CERR_ADDR*n*)

Access: User read/write

Address: 0h base + C24h offset + (8d × i), where i=0d to 19d



MEMU_FLASH_CERR_ADDR*n* field descriptions

Field	Description
0–31 ERR_ADD	Error address field Indicates the address on which the error was detected.

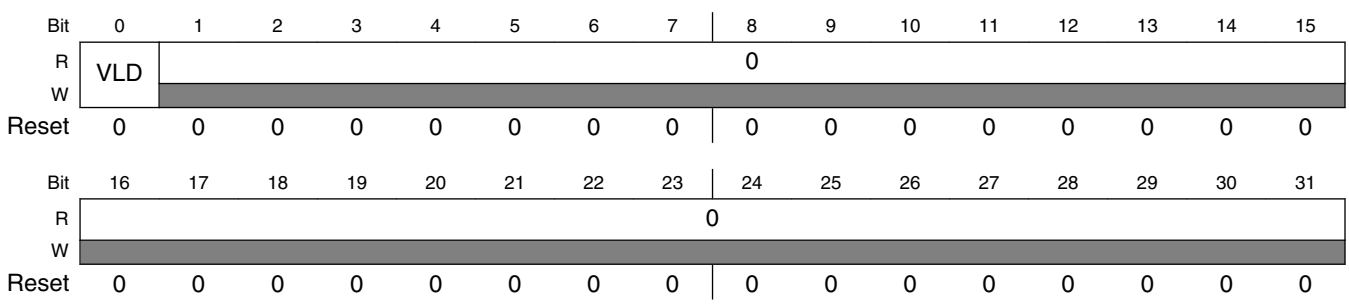
73.7.17 Flash memory uncorrectable error reporting table status register (MEMU_FLASH_UNCERR_STS)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC0h offset = CC0h



MEMU_FLASH_UNCERR_STS field descriptions

Field	Description
0 VLD	Valid bit. This indicates that the entry in reporting table is valid. Assertion of this bit causes the uncorrectable error detect flag to be asserted.

Table continues on the next page...

MEMU_FLASH_UNCERR_STS field descriptions (continued)

Field	Description
	0 Entry in table is invalid. 1 Entry in table is valid.
1–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

73.7.18 Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC4h offset = CC4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERR_ADD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_UNCERR_ADDR field descriptions

Field	Description
0–31 ERR_ADD	Error address field. Indicates the address on which the error was detected.

73.7.19 Flash memory concurrent overflow register (MEMU_FLASH_OFLW*n*)

Access: User read/write

NOTE

The reporting table is not cleared on software reset.

Address: 0h base + CC8h offset + (4d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OFLW																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MEMU_FLASH_OFLW*n* field descriptions

Field	Description
0–31 OFLW	Overflow Bit. Each bit corresponds to an error source and is asserted when any of the conditions mentioned in Handling overflows (Multiple error reporting) occurs.

73.8 Functional description

In this section, the term “bad bit” and “syndrome” are equivalent, and are used to indicate the position of the bit in error.

73.8.1 Initializing MEMU

MEMU is always enabled and can be configured by user software for known errors which the system/application has collected over a period of time. The MEMU logic will not update the reporting table corresponding to the programmed values which have the VLD bit SET.

The user software can write into the reporting table and must make VLD bit ='1' for the programmed entry to be valid. Any write by the CPU that asserts the VLD bit will cause a corresponding flag to FCCU.

The software can program the table with known error addresses by setting the valid bit and storing the corresponding error address. The sequence of programming to prevent a collision of reporting table entries written by the HW and the SW is as follows:

1. Check that the VLD bit is not set,
2. Write an invalid address/bad bit information,
3. Set VLD bit,
4. Check that the address still has an invalid value,
5. Write address/bad bit information to desired address.

If the FCCU is programmed to cause an IRQ on new MEMU table entries, steps 3 to 5 should be executed with interrupts disabled to prevent any error handling SW to read the invalid address written in step 2). Note that in the case the software is in between the 3rd and the 5th step, and the MEMU writes an error with the same address that Software wants to program, both the errors will be stored in the reporting table (i.e. duplicate entries).

73.8.2 Reading the reporting table

The software at any time can read the reporting table via the software programming interface. It should read the entries with the valid bit set and take necessary action. To invalidate any entry, the valid bit must be cleared. Reading an entry which does not have the VLD bit set will return invalid data.

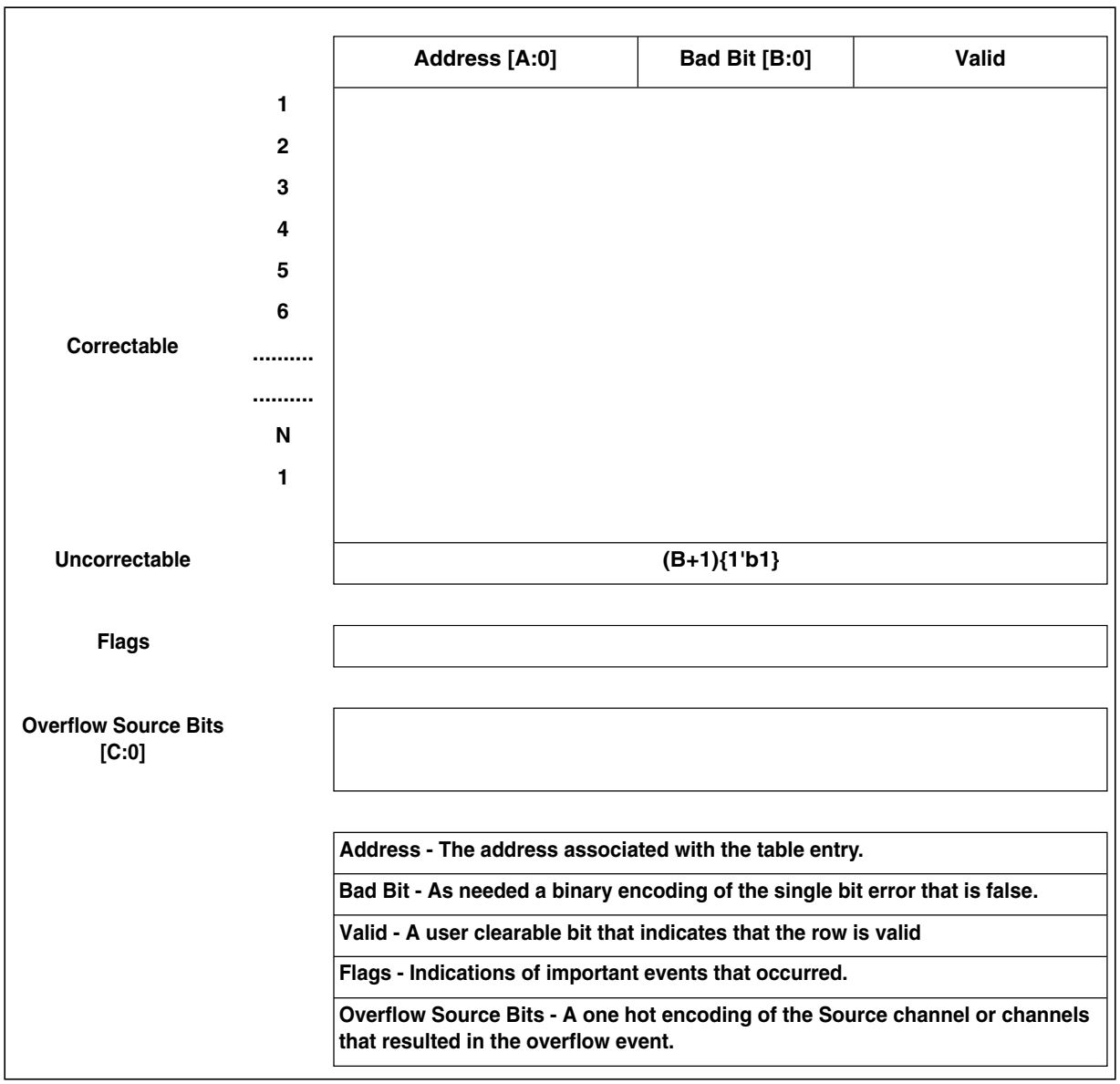


Figure 73-6. Generic representation of the reporting table of one MEMU reporting block

73.8.3 Handling overflows (Multiple error reporting)

When a bit in the overflow registers (either error buffer or reporting table though the overflow registers are same) is asserted it can indicate the occurrence of one of the following conditions:

1. Error buffer overflow–More than two memories reporting an error at the same instant or the input ASYNC FIFOs reach the FULL level (overflow)
    - More than 2 errors detected at same time. This would lead to overflow in the error buffer. MEMU can process only one error at a time while storing the other in the error buffer. So this would be indicated as an overflow. The uniqueness check in the input buffer does not "guarantee" that an ECC-supervised memory with only one error will not be marked as overflow source. If that one error occurs together with two (or more) additional errors at the same cycle it can get flagged as an overflow.
2. Correctable/Uncorrectable Error Overflow–Overflow in correctable and uncorrectable reporting table occurs only when a unique entry is to be stored but the tables are already full (all entries have valid bit set).
    - An ECC error which matches the address of an MBIST error is dropped (same as if it matched the address of an ECC error). In case, where a MBIST error comes with ECC error already in the table, both the address and bad bits are compared to determine uniqueness.
3. The Overflows will be stored in the bit-wise order i.e., if the error bit 31 reports an error, the bit 31 of the overflow register will be set to '1'.

Chapter 74
Fault Collection and Control Unit (FCCU)

74.1 Chip-specific FCCU information

74.1.1 FCCU configuration

This section describes the configuration of FCCU specific to the MPC5746C.

NOTE

To observed the toggling of the Eout signal as per protocol in fault state for FCCU, the SMC field of the MSCR register correspond to EOUT pads should be set.

NOTE

FCCU alarm will configure pads for tri-state unless SIUL2_MSCR[SMC] is enabled even though MCU is not in SAFE mode.

74.1.2 FCCU register reset values

All reset values for the FCCU registers that are device specific are shown in the following table. All other register reset values are shown in the FCCU memory map section.

Table 74-1. FCCU register reset values

Register	Register Value
Control Register (FCCU_CTRL)	0x0000_00C0
Configuration Register (FCCU_CFG)	See FCCU_CFG register bit value sources (N and C) by event.
NCF Configuration Register (FCCU_NCF_CFG0)	0xFFFF_FFFF
NCF Configuration Register (FCCU_NCF_CFG1)	0xFFFF_FFFF

Table continues on the next page...

Table 74-1. FCCU register reset values (continued)

Register	Register Value
NCF Configuration Register (FCCU_NCF_CFG2)	0x0000_00FF
NCFS Configuration Register (FCCU_NCFS_CFG0)	0x0008_2000
NCFS Configuration Register (FCCU_NCFS_CFG1)	0x0000_0000
NCFS Configuration Register (FCCU_NCFS_CFG2)	0x0000_0008
NCFS Configuration Register (FCCU_NCFS_CFG3)	0x0000_0000
NCFS Configuration Register (FCCU_NCFS_CFG4)	0x0000_0000
NCFS Configuration Register (FCCU_NCFS_CFG5)	0x0000_0000
NCF Enable Register (FCCU_NCF_E0)	0x0000_0240
NCF Enable Register (FCCU_NCF_E1)	0x0000_0002
NCF Enable Register (FCCU_NCF_E2)	0x0000_0000
NCF Time-out Enable Register (FCCU_NCF_TOE0)	0x0000_0000
NCF Time-out Enable Register (FCCU_NCF_TOE1)	0x0000_0000
NCF Time-out Enable Register (FCCU_NCF_TOE2)	0x0000_0000
MCS Register (FCCU_MCS)	0x0000_8083

74.1.3 FCCU_CFG register event bit values by source (N and C)

For each bit in the Configuration (FCCU_CFG) register, this table specifies the values of *N* and *C* for this chip. For information on how to interpret and use these values, see FCCU_CFG register bit value sources (N and C) by event.

Bit	N (chip-specific or user-specified value provided by the bit's associated NVMCFG signal)	C (chip-specific value provided by the bit's associated constant inside the module)
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	—	0
8	—	0
9	—	0
10	—	0
11	—	0

Table continues on the next page...

Bit	N (chip-specific or user-specified value provided by the bit's associated NVMCFG signal)	C (chip-specific value provided by the bit's associated constant inside the module)
12	—	0
13	—	0
14	—	0
15	—	0
16	0	0
17	0	0
18	0	0
19	—	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	0	0
26	0	0
27	0	0
28	0	1
29	0	0
30	0	0
31	0	0

74.1.4 Fault inputs

The following table shows the source of the fault signals and the type of fault input to which these signals are connected at the FCCU. NCF is used to identify a non-critical fault input at the FCCU.

Table 74-2. FCCU non-critical fault mapping

Non Critical Fault	Source	NCF_CFG (HW Rec/ SW Rec)	NCF_POL (Polarity of fault)	NCFS_CFG (Fault Reaction (Long/ Short Reset/ None))	NCF_EN (Enable fault source whether to go from state N to A or F)	NCF_TOEN (Time out EN or DIS by default)	Set/Clear available	Description
Register		**FCCU_NCF_CGF[0:3]**		**FCCU_NCFS_CFG[0:7]**	**FCCU_NCF_E[0:3]**	**FCCU_NCF_TOE[0:3]**		
NCF[0]	Flash	SW	HIGH	None	-	DIS	Yes	Flash reset error: Sideband signal from flash to indicate that flash encountered faults during its reset reads. This is caused by ECC double bit detection as well as the coherency checks done on the test row reads.
NCF[1]	Flash	SW	HIGH	None	-	DIS	Yes	Flash Low power error: A signal to indicate that a request for Flash entry into Low Power mode was rejected. This occurs if a LP request is sent if a High Voltage or Utest operations are ongoing.
NCF[2][1]	Flash	SW	HIGH	None	-	DIS	Yes	Flash read voltage error: A fault signal indication from flash, it tells that previous reads were affected by a read voltage loss of regulation.
NCF[3][1]	Flash	SW	HIGH	None	-	DIS	Yes	Flash reference current error:

Table continues on the next page...

Table 74-2. FCCU non-critical fault mapping (continued)

Non Critical Fault	Source	NCF_CFG (HW Rec/ SW Rec)	NCF_POL (Polarity of fault)	NCFS_CFG (Fault Reaction (Long/ Short Reset/ None))	NCF_EN (Enable fault source whether to go from state N to A or F)	NCF_TOEN (Time out EN or DIS by default)	Set/Clear available	Description
								A fault signal indication from flash, it tells that previous reads were affected by a read current loss of regulation.
NCF[4]	SSCM	SW	HIGH	None	-	DIS	-	Memories DCF client safety error (BIST)
NCF[5]	SSCM	SW	HIGH	None	-	DIS	-	Configuration error during reset while loading STCU configuration
NCF[6]	STCU	SW	HIGH	Long Reset	Yes	DIS	Yes	BIST result error # critical fault
NCF[7]	STCU	SW	HIGH	None	-	DIS	Yes	BIST result error # non-critical fault
NCF[8]	STCU	SW	HIGH	None	-	DIS	Yes	Critical fault indication from STCU in case selftest control signals go to wrong condition during user application
NCF[9]								Reserved
NCF[10]	ADC0 CF	SW	HIGH	None	-	DIS	Yes	ADC0 Internal Selftest critical fault,
NCF[11]	ADC0 NCF	SW	HIGH	None	-	DIS	Yes	ADC0 Internal Selftest non-critical fault,
NCF[12]								Reserved
NCF[13]	SWT_1	SW	HIGH	None	-	DIS	-	SWT_1 raw interrupt request
NCF[14]	SWT_0	SW	HIGH	None	-	DIS	-	SWT_0 raw interrupt request
NCF[15]								Reserved
NCF[16]	MEMU_0	SW	HIGH	None	-	DIS	-	System RAM correctable ECC error
NCF[17]	MEMU_0	SW	HIGH	None	-	DIS	-	System RAM uncorrectable ECC error
NCF[18]	MEMU_0	SW	HIGH	None	-	DIS	-	System RAM error overflow (all or'ed)

Table continues on the next page...

Table 74-2. FCCU non-critical fault mapping (continued)

Non Critical Fault	Source	NCF_CFG (HW Rec/ SW Rec)	NCF_POL (Polarity of fault)	NCFS_CFG (Fault Reaction (Long/ Short Reset/ None))	NCF_EN (Enable fault source whether to go from state N to A or F)	NCF_TOEN (Time out EN or DIS by default)	Set/Clear available	Description
NCF[19]	MEMU_0	SW	HIGH	None	-	DIS	-	Peripheral RAM correctable ECC error
NCF[20]	MEMU_0	SW	HIGH	None	-	DIS	-	Peripheral RAM uncorrectable ECC error
NCF[21]	MEMU_0	SW	HIGH	None	-	DIS	-	Peripheral RAM error overflow (all or'ed)
NCF[22]	MEMU_0	SW	HIGH	None	-	DIS	-	Flash correctable ECC error
NCF[23]	MEMU_0	SW	HIGH	None	-	DIS	-	Flash uncorrectable ECC error
NCF[24]	MEMU_0	SW	HIGH	None	-	DIS	-	Flash Error Overflow (all or'ed)
NCF[25] [2], [1]	PRAM_0	SW	HIGH	None	-	DIS	-	System PRAM Controller_0 ahb alarm
NCF[26][2, 1]	PRAM_ 1	SW	HIGH	None	-	DIS	-	System PRAM Controller_1 ahb alarm
NCF[27]								Reserved
NCF[28]	MC_CGM	SW	HIGH	None	-	DIS		sys clk mux glitchless functionality failed
NCF[29]	ADC0_WD	SW	HIGH	None	-	DIS	-	ADC Watchdog Interrupt
NCF[30]	ADC1_WD	SW	HIGH	None	-	DIS	-	ADC Watchdog Interrupt
NCF[31]					-		-	Reserved
NCF[32]	CMU0 [3]	SW	HIGH	None	-	DIS		PLL frequency out of range
NCF[33]	FM_PLL_0 [3]	SW	HIGH	Long reset	Yes	DIS	-	Loss of lock
NCF[34][1]	SMPU0	SW	HIGH	None	-	DIS		SMPU FCCU Alarm
NCF[35]								Reserved
NCF[36][1]	PFLASH Controller	SW	HIGH	None	-	DIS	Yes	Error detected in address manipulation (logic through EDC)
NCF[37][1]	PFLASH Controller	SW	HIGH	None	-	DIS	only clr	pfl_edc_fccu_alarm error in ECC Correction (logic through EDC)

Table continues on the next page...

Table 74-2. FCCU non-critical fault mapping (continued)

Non Critical Fault	Source	NCF_CFG (HW Rec/ SW Rec)	NCF_POL (Polarity of fault)	NCFS_CFG (Fault Reaction (Long/ Short Reset/ None))	NCF_EN (Enable fault source whether to go from state N to A or F)	NCF_TOEN (Time out EN or DIS by default)	Set/Clear available	Description
NCF[38][1]	PFLASH Controller	SW	HIGH	None	-	DIS	only clr	fl_enc_fccu_alarm error in ECC Correction (single bit inversions are detected repaired but not reported)
NCF[39][1]	PFLASH Controller	SW	HIGH	None	-	DIS	only clr	pfl_enc_fccu_alarm error in ECC Correction (single bit inversions are detected repaired but not reported)
NCF[40][1]	HSM-SMPU	SW	HIGH	None	-	DIS		HSM SMPU FCCU Alarm
NCF[41]					-			Reserved
NCF[42][1]	PFLASH Controller	SW	HIGH	None	-	DIS	only clr	fl_safe_cal_fccu_alarm alarm. (Error detected in transaction monitoring of the pseudo replicated calibration evaluation logic)
NCF[43][1]	PRAM_0	SW	HIGH	None	-	DIS	only clr	System PRAM Controller_0 RAM alarm
NCF[44][1]	PRAM_ 1	SW	HIGH	None	-	DIS	only clr	System PRAM Controller_1 ram alarm
NCF[45]					-			Reserved
NCF[46]	Internal	SW	HIGH	None	-	DIS	-	-
NCF[47]	Internal	SW	HIGH	None	-	DIS	-	-
NCF[48]	Internal	SW	HIGH	None	-	DIS	-	-
NCF[49]	Internal	SW	HIGH	None	-	DIS	-	-
NFC[50]	RGM	SW	HIGH	None	-	DIS	-	Safe Mode requested
NCF[51][1]	Core- z4	SW	HIGH	None	-	DIS	-	machine check out
NCF[52]								Reserved
NCF[53][1]	Core-Z2	SW	HIGH	None	-	DIS	-	machine check out
NCF[54]	EIN_ERR(from PAD)	SW	HIGH	None	-	DIS		Error input pin (from the external world)
NCF[55][1]	Core-HSM	SW	HIGH	None	-	DIS		machine check out
NCF[56]	MEMU_1	SW	HIGH	None	-	DIS		System RAM correctable ECC error

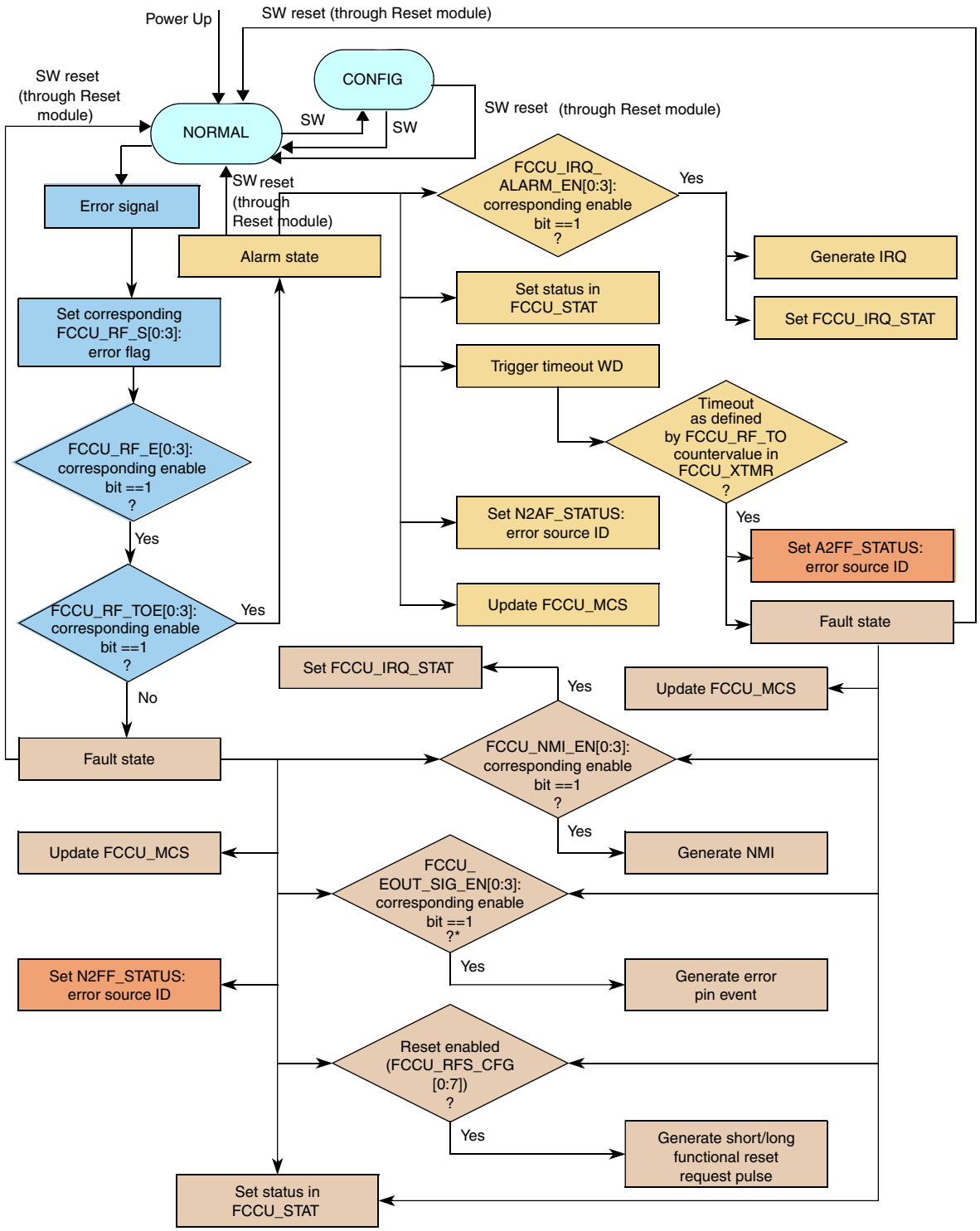
Table continues on the next page...

Table 74-2. FCCU non-critical fault mapping (continued)

Non Critical Fault	Source	NCF_CFG (HW Rec/ SW Rec)	NCF_POL (Polarity of fault)	NCFS_CFG (Fault Reaction (Long/ Short Reset/ None))	NCF_EN (Enable fault source whether to go from state N to A or F)	NCF_TOEN (Time out EN or DIS by default)	Set/Clear available	Description
NCF[57]	MEMU_1	SW	HIGH	None	-	DIS		System RAM uncorrectable ECC error
NCF[58]	MEMU_1	SW	HIGH	None	-	DIS		System RAM error overflow
NCF[59]	MEMU_1	SW	HIGH	None	-	DIS		Peripheral RAM correctable ECC error
NCF[60]	MEMU_1	SW	HIGH	None	-	DIS		Peripheral RAM uncorrectable ECC error
NCF[61]	MEMU_1	SW	HIGH	None	-	DIS		Peripheral RAM error overflow
NCF[62]	MBIST Error	SW	HIGH	None	-	DIS		MBIST Error PRAM_0
NCF[63]	MBIST Error	SW	HIGH	None	-	DIS		MBIST Error PRAM_0, PRAM_1
NCF[64]	MBIST Error	SW	HIGH	None	-	DIS		MBIST ERROR z4 (DTAG, DCACHE)
NCF[65]	MBIST Error	SW	HIGH	None	-	DIS		MBIST ERROR z4 (ITAG, ICACHE)
NCF[66]	MBIST Error	SW	HIGH	None	-	DIS		MBIST ERROR HSM (TAG, DCACHE, RAM)
NCF[67]	MBIST Error	SW	HIGH	None	-	DIS		MBIST ERROR CAN
NCF[68]	MBIST Error	SW	HIGH	None	-	DIS		MBIST ERROR Non-Core masters (except HSM)
NCF[69]	ADC1 CF	SW	HIGH	None	-	DIS		ADC1 Internal Selftest critical fault
NCF[70]	ADC1 NCF	SW	HIGH	None	-	DIS		ADC1 Internal Selftest non-critical fault
NCF[71]					-			Reserved

1. These faults may get set during chip level debug mode and needs to be ignored.
2. The FCCU alarms indicate a fault in the PRAMC internal address decode that causes an errant hit/miss evaluation of the PRAMC store buffer.
3. This source should be configured to generate reset when the respective clocks are used as system clocks.

74.1.5 Error signal flow diagram



* This condition must be satisfied only when FCCU is configured for Bistable fault-output mode (FCCU_CFG[FOM]).

Figure 74-1. Error signal flow

74.2 Introduction

The Fault Collection and Control Unit (FCCU) offers a hardware channel to collect faults and to place the device into a safe state when a failure in the device is detected. No CPU intervention is requested for collection and control operation.

The FCCU offers a systematic approach to fault collection and control. The distinctive features of the module are:

- Redundant collection of fault information from safety relevant modules on the device
- Collection of test results
- Configurable fault control
- Configurable internal reactions for each NCF:
  - No reaction
  - IRQ
  - Short functional reset
  - Long functional reset
  - NMI
- External reactions via configurable output pins
- Watchdog timer for the reconfiguration phase
- Lockable configuration
- Nonvolatile memory interface (NVMCFG signals) for configuration loading
- One of the fault-output (EOUT) signals is high to indicate operational (OK) state (in bi-stable operation mode). The above is not true in case the error out protocol is configured to be a toggling protocol, for example, dual rail and time switching.
- After power on, the EOUT signals have high impedance.[1] They show operational state only after configuration by software.
- In case of a failure event or on software request for Error pin indication, the pin(s) are set to faulty state for a minimum time T_min, even if SW tries to release it before (for the case of error pin configured in Bi-stable mode only).

1. Actual value depends on device specific setting at pad level.

- The overall fault detection, processing and indication time is less than 10 ms.
- The actual maximum time is 5 CLKSAFE cycles**.**

The FCCU is operative with the default configuration. Internal (short or long functional reset request pulse, interrupt request) and external (EOUT signaling) reactions are statically defined or programmable. The default configuration can be modified only in the CONFIG state. FCCU is designed to function when CLKSYS is faster than the CLKSAFE clocks.

74.2.1 Acronyms and abbreviations

Term	Description
CF	Critical fault
CPU	Central processing unit
EOUT	Error out
FSM	Finite state machine
INTC	Interrupt controller
intf	Interface
IRQ	Interrupt request
MC_ME	Mode entry module
NCF	Noncritical fault
NMI	Nonmaskable interrupt
NVM	Nonvolatile memory
SoC	System-on-chip
STCU	Self-test control unit
SW	Software
WKPU	Wake-up unit

74.3 Main features

- Management of noncritical faults
- HW or SW fault recovery management
- Fault detection and collection
- Fault injection (fake faults)

- External reaction (fault state): EOUT signaling. Error indication via the pin(s) is controlled by the FCCU.
- Internal chip reactions (alarm state): interrupt request
- Internal chip reactions (fault state):
  - long functional reset request pulse
  - short functional reset request pulse
  - NMI
- Bi-Stable, Dual-Rail and Time Switching output protocols on EOUT
- Internal (to the FCCU) watchdog timer for the reconfiguration phase
- Configuration lock
- Nonvolatile memory interface (NVMCFG signals) for configuration loading

74.4 Block diagram

The following figure is a top-level diagram of the FCCU module.

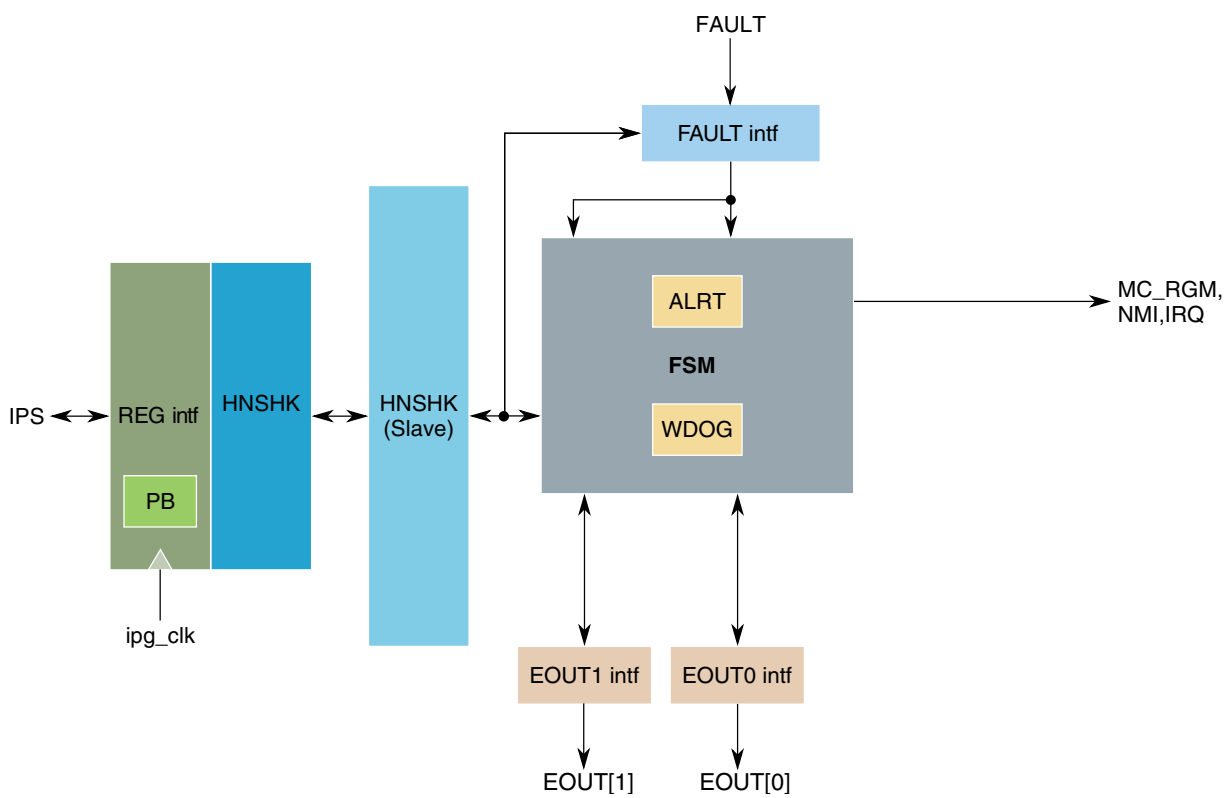


Figure 74-2. FCCU block diagram

This table describes the FCCU submodules.

Table 74-3. FCCU submodules

Submodule	Description
REG intf	Includes the register file, the IPS bus interface, the IRQ interface and the parity block (PB) for the configuration registers
HNSHK blocks (master and slave blocks)	Includes the FSM's ability to support the handshake between the REG interface and the FSM unit due to the usage of 2 asynchronous clocks (IPS system clock and RC oscillator clock)
FSM unit	Implements the main functions of the FCCU. The FSM also includes the: • Watchdog timer (WDG) • Alarm timer (ALRT)
FAULT intf	Implements the interface for the fault conditioning and management
EOUTx units	Implement the output stage to manage the EOUT interfaces

74.5 Signal description

74.5.1 Signals

This table describes the signals on the boundary of FCCU.

Signal	Direction	Function	Active level	Clock
Clock and reset				
CLKSYS	Input	System Clock—Supplies the primary clock.	Rising edge	—
CLKSAFE0	Input	Safe Clock 0—Supplies one of two 16 MHz clocks for use by various redundant submodules. This clock and CLKSAFE1 are often referred to as the same clock (CLKSAFE).	Rising edge	—
CLKSAFE1	Input	Safe Clock 1—Supplies one of two 16 MHz clocks for use by various redundant submodules. This clock and CLKSAFE0 are often referred to as the same clock (CLKSAFE).	Rising edge	—
EOUT interface				
EIN[1:0]	Input	Error Input—Provide the mechanism for FCCU to capture the states that off-chip logic drives on the associated EOUT[1:0] signals. (On the chip, the EOUT signals are connected to their corresponding EIN signals.) FCCU captures the states in the EIN1 and EIN0 fields.	—	Asynchronous

Table continues on the next page...

Signal	Direction	Function	Active level	Clock
EOUT[1:0]	Output	Error Output—Indicate FCCU's condition (Faulty, Nonfaulty, or Configuration) to off-chip logic.	Programmable	CLKSAFE0, CLKSAFE1
WKPU interface				
NMIWKPU	Input	WKPU Nonmaskable Interrupt Request—Receives nonmaskable interrupt requests from the WKPU module.	Low	CLKSYS
WAKEWKPU	Output	Wake WKPU—Wakes the WKPU module to bring the chip out of dormant state when FCCU enters Fault state as the result of a noncritical fault for which NMI is enabled as the reaction (FCCU_NMI_ENa[NMIENx]).	High	CLKSAFE0, CLKSAFE1
Interrupt interface				
NMIOUT	Output	Nonmaskable Interrupt Output—Sends a nonmaskable interrupt request to the processor core or cores when FCCU enters Fault state as the result of a noncritical fault for which NMI is enabled as the reaction (FCCU_NMI_ENa[NMIENx]).	Low	CLKSYS
NVM interface				
NVMCFG[21:0]	Input	Nonvolatile Memory Configuration—Provide the values of certain bits in the Configuration (FCCU_CFG) register when certain events occur. See Configuration (FCCU_CFG). Some or all of these signals may be driven by bits in a data record in nonvolatile (flash) memory or may be tied high (1) or low (0). See FCCU_CFG register bit value sources (N and C) by event.	Static value	—
SSCMDONE	Input	SSCM Done—After an FCCU reset, indicates that the SSCM module is now driving any NVMCFG signals that are associated with a data record stored in nonvolatile (flash) memory.	High	CLKSYS

74.6 Put FCCU in Configuration or Normal state

74.6.1 Introduction

You put FCCU in Configuration state to change its configuration. You put FCCU in Normal state to save changes to the configuration.

74.6.2 About changing states

When changing states, keep the following in mind:

- To put FCCU in Configuration state, FCCU must be in Normal state.
- After FCCU is reset, the configuration is temporarily locked. You must temporarily unlock the configuration before you can put FCCU in Configuration state.
- When FCCU is in Configuration state, FCCU doesn't actually save the changes you make to the configuration. To save changes to the configuration, you must manually put FCCU in Normal state. If FCCU automatically leaves Configuration state and enters Normal state because the configuration-timer interval (TO) expires (called a Configuration-state timeout), FCCU changes the value of the Configuration (FCCU_CFG) register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. For information on the Configuration-state timeout value, see FCCU_CFG register bit value sources (N and C) by event. For a list of configuration registers, see Configuration registers.

74.6.3 Put FCCU in Configuration state

1. Temporarily unlock the configuration (see Temporarily unlock the configuration).
2. To determine FCCU's state, run the OP3 operation (see Run an operation).
3. Check the operation status (OPS).
   - If the operation status is Successful, go to step 4.
   - If the operation status is Aborted, go to step 2.
4. Check the FCCU status (STATUS).
   - If the FCCU status is Configuration, FCCU is in Configuration state. Stop.
   - If the FCCU status is Normal, go to step 5.
   - If the FCCU status is Alarm or Fault, go to step 7.
5. To put FCCU in Configuration state, run the OP1 operation (see Run an operation).
6. Check the operation status (OPS).
   - If the operation status is Successful, FCCU is in Configuration state. Stop.
   - If the operation status is Aborted, the configuration is probably permanently locked. Go to step 9.
7. Recover all faults.
8. Go to step 2.
9. Reset FCCU.

74.6.4 Put FCCU in Normal state

1. To determine FCCU's state, run the OP3 operation (see Run an operation).
2. Check the operation status (OPS).
   - If the operation status is Successful, go to step 3.
   - If the operation status is Aborted, go to step 1.
3. Check the FCCU status (STATUS).
   - If the FCCU status is Normal, FCCU is in Normal state. Go to step 8.
   - If the FCCU status is Configuration, go to step 4.
   - If the FCCU status is Alarm or Fault, go to step 6.
4. To put FCCU in Normal state, run the OP2 operation (see Run an operation).
5. Check the operation status (OPS).
   - If the operation status is Successful, FCCU is in Normal state. Go to step 8.
   - If the operation status is Aborted, go to step 4.
6. Recover all faults.
7. Go to step 1.
8. If you want to prevent changes to the configuration, lock it:
   - To require a key to unlock the configuration, temporarily lock the configuration (see Temporarily lock the configuration).
   - To require a reset of FCCU to unlock the configuration, permanently lock the configuration (see Permanently lock the configuration).

74.7 Run operations

74.7.1 Introduction

You run operations to perform actions such as putting FCCU in Configuration state or reading a timer. For a complete list of operations you can run, see OPR.

74.7.2 About running operations

When running operations, keep the following in mind:

- FCCU ignores any operations initiated while the operation status is In Progress.
- Certain operations must be unlocked before you can initiate them. After you initiate them, they are again locked.

74.7.3 Run an operation

1. Check the operation status (OPS).
2. If the operation status is In Progress, go to step 1.
3. If the operation must be unlocked before you can initiate it, unlock the operation (CTRLK) using a 32-bit write.

NOTE

The Control Key (FCCU_CTRLK) register used in this step and the Control (FCCU_CTRL) register used in the next step must be written with consecutive instructions. Do not use read-modify-write instructions, such as bit-field instructions, to modify these registers.

4. Initiate the operation (OPR) using a 32-bit write.
5. Check the operation status (OPS).
6. If the operation status is In Progress, go to step 5.

The operation status is now Idle (OP15 only), Aborted, or Successful.

74.8 Lock and unlock the configuration

74.8.1 Introduction

After you configure FCCU, you can lock its configuration to prevent accidental changes.

74.8.2 About locking the configuration

When locking the configuration, keep the following in mind:

- If you attempt to lock the configuration while FCCU is in Configuration state, FCCU doesn't actually lock the configuration until FCCU leaves Configuration state—that is, either you put FCCU in Normal state, or FCCU puts FCCU in Normal state because the configuration-timer interval (TO) expires.
- After you permanently lock the configuration, you must reset FCCU before you can put FCCU in Configuration state.

74.8.3 Temporarily lock the configuration

1. From Supervisor mode, write a lock value to TRANSKEY.
2. Put FCCU in Normal state if it's in Configuration state (see Put FCCU in Normal state).

74.8.4 Temporarily unlock the configuration

- From Supervisor mode, write the unlock value to TRANSKEY.

74.8.5 Permanently lock the configuration

1. From Supervisor mode, write the lock value to PERMNTKEY.
2. Put FCCU in Normal state if it's in Configuration state (see Put FCCU in Normal state).

The configuration is permanently locked until FCCU is reset.

74.9 Functional description

74.9.1 Definitions

In general, the following definitions are applicable for the fault management:

- HW recoverable fault: the fault indication is an edge-triggered and level-sensitive signal that remains asserted until the fault cause is deasserted. That is, if logical 0 on the fault signal indicates fault, then the status flags are valid as long as the fault line stays at 0. The status is automatically cleared when the fault signal goes to 1. Typically the fault signal is latched external to the FCCU in the module where the fault occurred. The FCCU state transitions are consequently executed on the state changes of the input fault signal. No SW intervention in the FCCU is required to recover the fault condition.
- SW recoverable fault: the fault indication is a signal asserted without a defined time duration. The fault signal is latched in the FCCU. The fault recovery is executed following a SW recovery procedure (status/flag register clearing).

HW recoverable is an option to exclude the handling of error source/s by FCCU management SW, in case it is known that the fault is recoverable by itself when the fault condition gets corrected.

These type of chip resets reset the FCCU:
- Power-on
- Destructive
- Long functional, when initiated by the assertion of the chip reset pin (RESET_B)

These types of chip resets don't reset the FCCU:
- Long functional, when initiated by anything other than the assertion of the chip reset pin (RESET_B)
- Short functional reset

For more information on each type of reset, see the MC_RGM and reset chapters.

74.9.2 FSM description

FCCU functionality is depicted by the finite state machine (FSM) state diagram.

Basically four states are identified with the following meaning:

- Configuration: Used only to modify the configuration of FCCU from its default. A subset of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, timeout, noncritical fault masking) can be accessed in write mode only in Configuration state.

  Configuration state is accessible only in Normal state and if the configuration is not locked. The permanent configuration lock can be disabled by a reset which also resets the FCCU., The transient lock register is unlocked by writing BCh into it. FCCU gets transiently locked again if an invalid key is written into FCCU_TRANS_LOCK register (that is, other than BCh). Key to lock FCCU permanently is FFh, written in the FCCU_PERMNT_LOCK register.

  After the release of reset the state of the transient lock will be locked. The state of the permanent lock will be unlocked.

  The locking feature only restricts the FSM movement into Configuration state. Once the user enters Configuration state and then tries to lock the configuration, the locking of configuration will be effective after FCCU moves to Normal state, it will not be effective in the current Configuration state.

  The Configuration to Normal state transition can be executed by SW or automatically following a timeout condition of the watchdog. In case the timeout information and the SW request to mode change to Normal appears at the same time, watchdog

timeout has the priority and hence the Configuration registers (those that are writable only in Configuration state) are reset to their default values. The movement to Normal is made.

The incoming faults, occurring during the configuration phase (Configuration state) are latched in order to process them when FCCU is moved into Normal state, according to the new configuration.

All pending faults that occur during Configuration state result in both of the following:

- Highest-priority state transition
- Interrupt generation (NMI or alarm IRQ)

If the state transition occurs, it gives the reset reaction corresponding to the worst case based on all the faults (pending or nonpending faults) that occurred during Configuration state.

- Normal: This is FCCU's operating state when no faults are occurring. It is also the default state on the reset exit. Following state transitions occur on one of the following events:
  - unmasked noncritical faults with the timeout disabled → FCCU moves to Fault state
  - unmasked noncritical faults with the timeout enabled → FCCU moves to Alarm state
  - masked noncritical faults → FCCU stays in Normal state
- Alarm: FCCU moves into Alarm state when an unmasked noncritical fault occurs and the timeout is enabled. The transition to Alarm state goes along with an interrupt alarm, if enabled. By definition, this fault may be recovered within a programmable timeout period, before it generates a transition to Fault state. The timeout is reinitialized if FCCU enters Normal state. The timeout restarts following the recovery from Fault state.
- Fault: FCCU moves into Fault state when one of the following condition occurs:
  - timeout related to a noncritical fault when FCCU is in Alarm state
  - unmasked noncritical faults with the timeout disabled

The transition from Normal/Alarm to Fault state goes along with the generation of:

- NMI interrupt (optional)

- EOUT signaling (optional)
- SW option: Soft reaction (Short functional reset request pulse if configured)
- SW option: Hard reaction (Long functional reset request pulse if configured)

After moving to Fault state, if there is either a previous pending fault or a new fault for which NMI is enabled, NMI generation takes place.

Multiple faults can occur at the same time. Consider the case where there is a fault for which Alarm state is enabled and IRQ is programmed along with a fault for which Alarm state is disabled. In that case, the FSM moves to Fault state, but with the generation of alarm IRQ for the fault which has IRQ enabled.

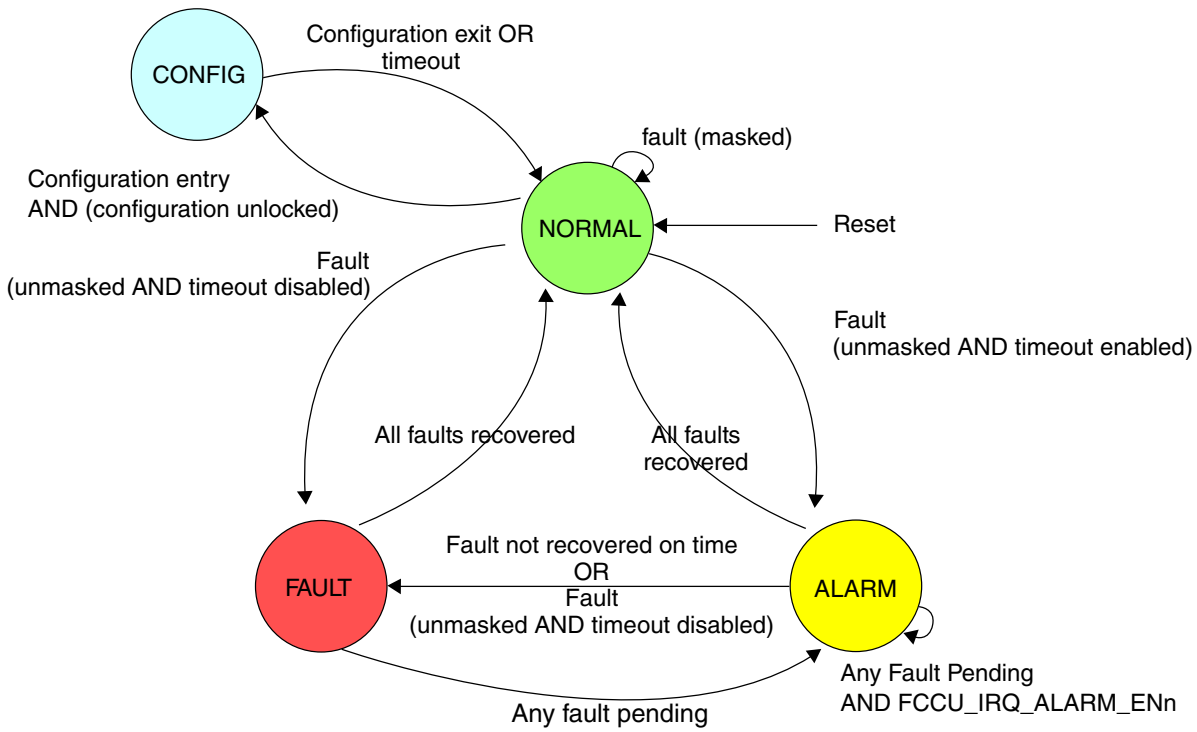


Figure 74-3. FCCU state diagram

74.9.3 Fault priority scheme and nesting

The FAULT state has a higher priority than the ALARM state in case of concurrent fault events (noncritical) that occur in the NORMAL state.

The ALARM to FAULT state transition occurs if a noncritical fault (unmasked and with time-out disabled) is asserted in the ALARM state.

The ALARM to NORMAL state transition occurs only if all the noncritical faults (including the faults that have been collected after the entry in the ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will remain in the ALARM state.

The FAULT to NORMAL state transition occurs only if all the noncritical faults (including the faults that have been collected after the entry in the FAULT/ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU will move to the ALARM state (if any noncritical fault is still pending and the time-out is not elapsed).

In general, no fault nesting is supported except for the noncritical faults that cause an ALARM to FAULT state transition. In this case the NCF timer is stopped until the FAULT state is recovered. If FCCU is in ALARM state and another fault comes, which has its alarm timeout enabled, then the alarm timer shall not reload and shall not start again.

74.9.4 Fault recovery

The following timing diagrams describe the main use cases of the FCCU in terms of fault events and related recovery.

A typical sequence related to a noncritical FAULT management (ALARM state), see Figure 74-4 and Figure 74-5, is as follows:

- Noncritical fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - alarm interrupt request (if enabled)
  - time-out running
- system state: RUN
- alarm interrupt management: FAULT recovery (by SW)
  - FCCU state transition ALARM → NORMAL

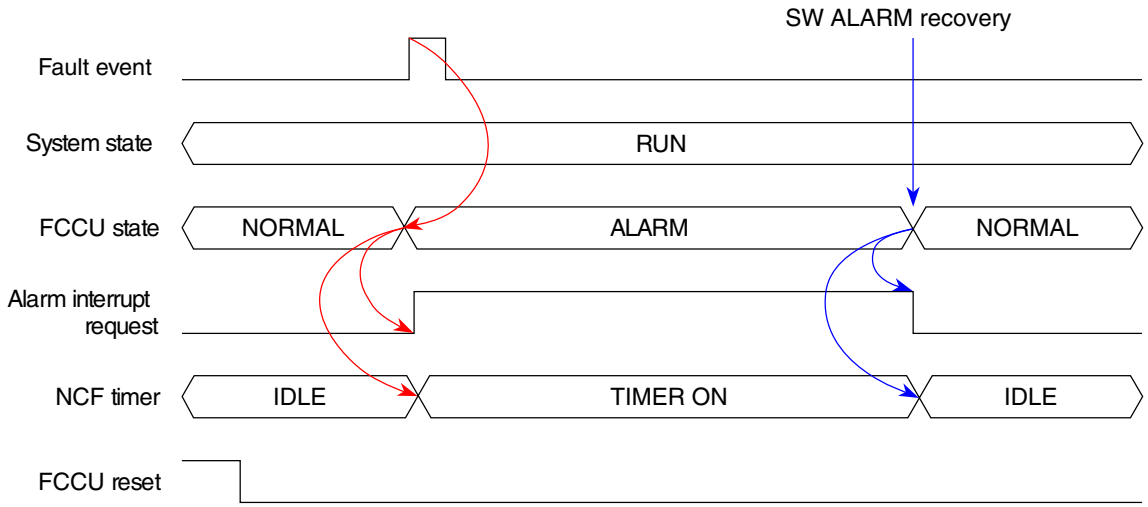


Figure 74-4. Noncritical FAULT (ALARM state) recovery (a)

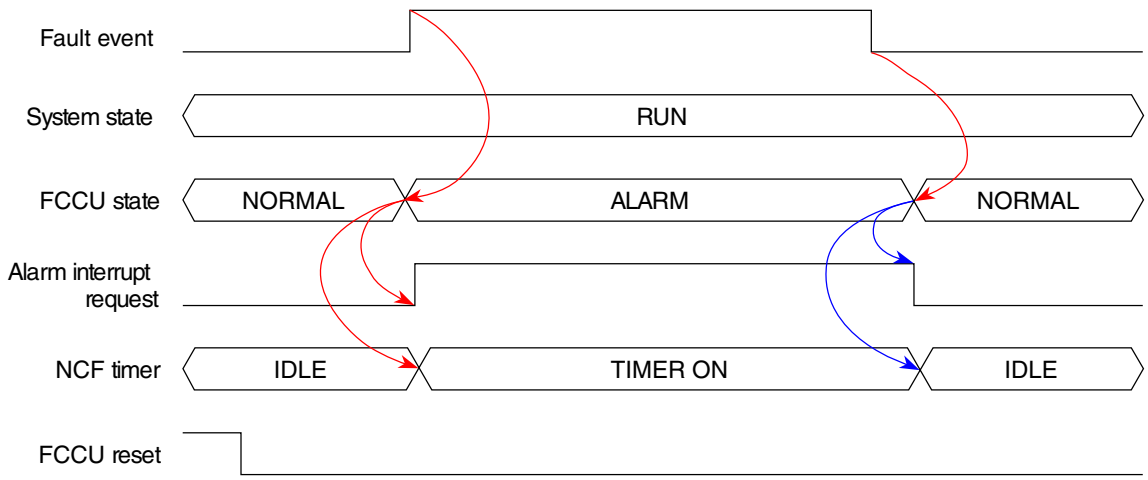


Figure 74-5. Noncritical FAULT (ALARM state) recovery (b)

A typical sequence related to a noncritical FAULT management (ALARM → FAULT state), see Figure 74-6, is as follows:

- Noncritical fault assertion
- FCCU state transition (automatic): NORMAL → ALARM
  - alarm interrupt request (if enabled)
  - time-out running
- FCCU state transition (following the time-out trigger): ALARM → FAULT
  - NMI assertion (if enabled)
- system state transition: RUN → SAFE
- NMI interrupt management (if enabled)

- FAULT recovery (by SW): FCCU state transition FAULT → NORMAL
- system state transition: SAFE → RUN

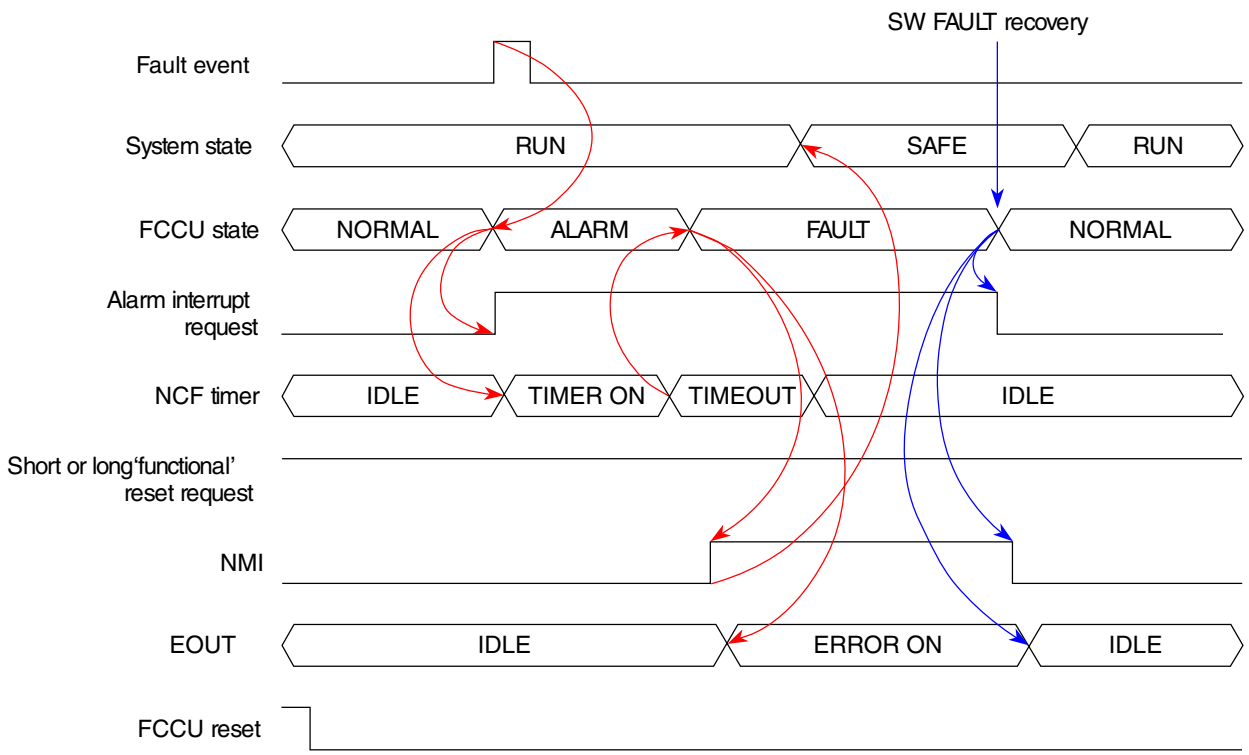


Figure 74-6. FAULT (ALARM → FAULT state) recovery

74.9.5 NMI/WKPU interface

NMIOUT is connected to all processor cores on the chip but is masked when any one of the following is true:

- An FCCU asynchronous reset is in progress.
- The chip is in RESET mode.
- The chip has automatically entered DRUN mode from RESET mode and software has not yet requested another mode change.

This figure shows the logical scheme.

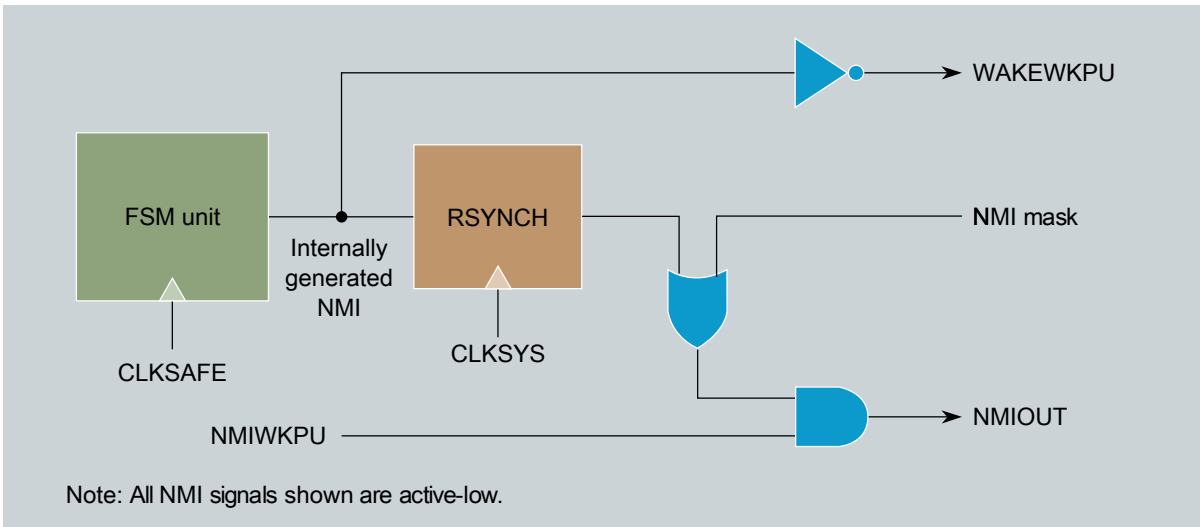


Figure 74-7. NMI/WKPU scheme

74.9.6 STCU interface

The STCU interface includes:

- A set of signals resulting from the self-checking procedure connected externally at the FCCU.

  The STCU fault signals are processed by the FCCU when the SoC is re-booted following the self-testing procedure. The STCU includes a status register that stores the self-testing results (flags).

- a mask that inhibits the EOUT dummy signaling until the STCU self-checking procedure has been completed.

During the self-testing procedure, depending on the STCU results, three cases are applicable:

1. STCU completes the self-testing procedure successfully. The SoC reboots and the FCCU is responsible for fault collection and indication.
2. STCU completes the self-testing procedure with low severity failures. The FCCU records the fault condition but does not, by default, trigger a reset cycle.
3. STCU completes the self-testing procedure with serious failures. The FCCU triggers a reset cycle in response to this fault input from the STCU. Because the fault condition survives reset, new reset cycles are subsequently triggered until the reset escalation feature takes control, freezing the chip in the reset state until a POR. This feature is optionally programmable using MC_RGM.

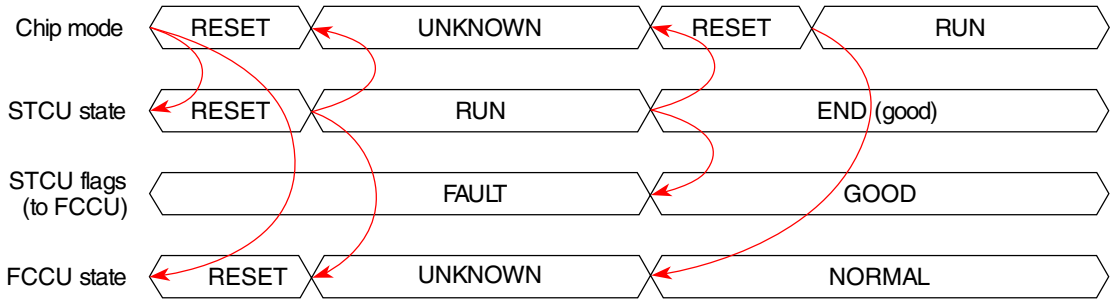


Figure 74-8. STCU-FCCU (case 1)

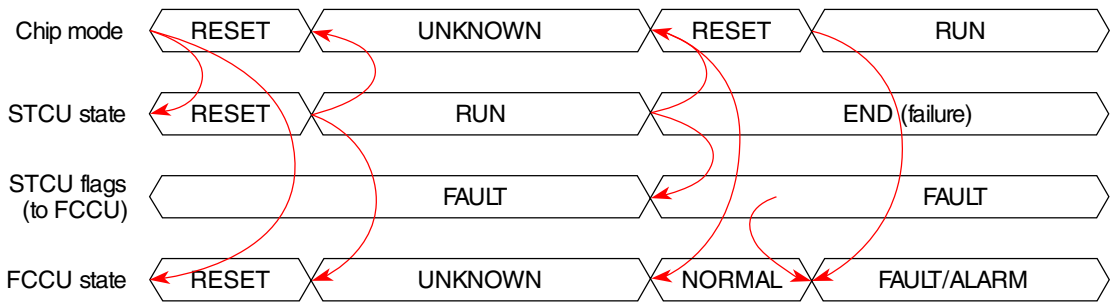


Figure 74-9. STCU-FCCU (case 2)

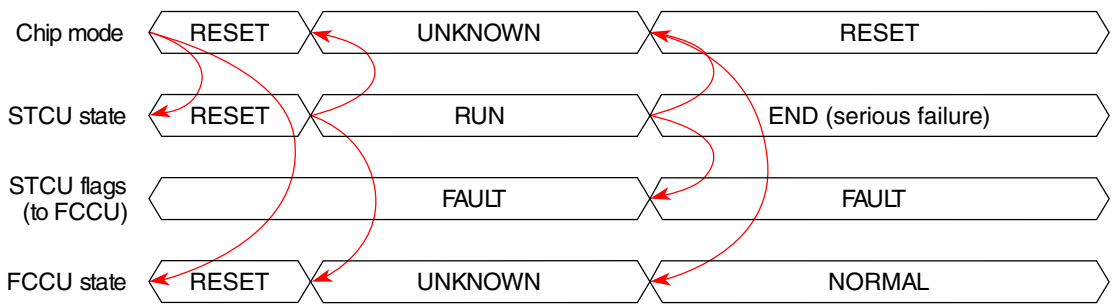


Figure 74-10. STCU-FCCU (case 3)

74.9.7 Nonvolatile memory interface

FCCU includes a nonvolatile memory interface (NVMCFG signals) that provides the values of certain bits in the Configuration (FCCU_CFG) register when certain events occur. See Configuration (FCCU_CFG). Whether any of these signals are actually connected to nonvolatile (flash) memory is chip-specific. Some or all of these signals may be driven by bits in a data record in nonvolatile (flash) memory or may be tied high (1) or low (0). See FCCU_CFG register bit value sources (N and C) by event.

The timing diagram related to the NVM interface is given in Figure 74-11.

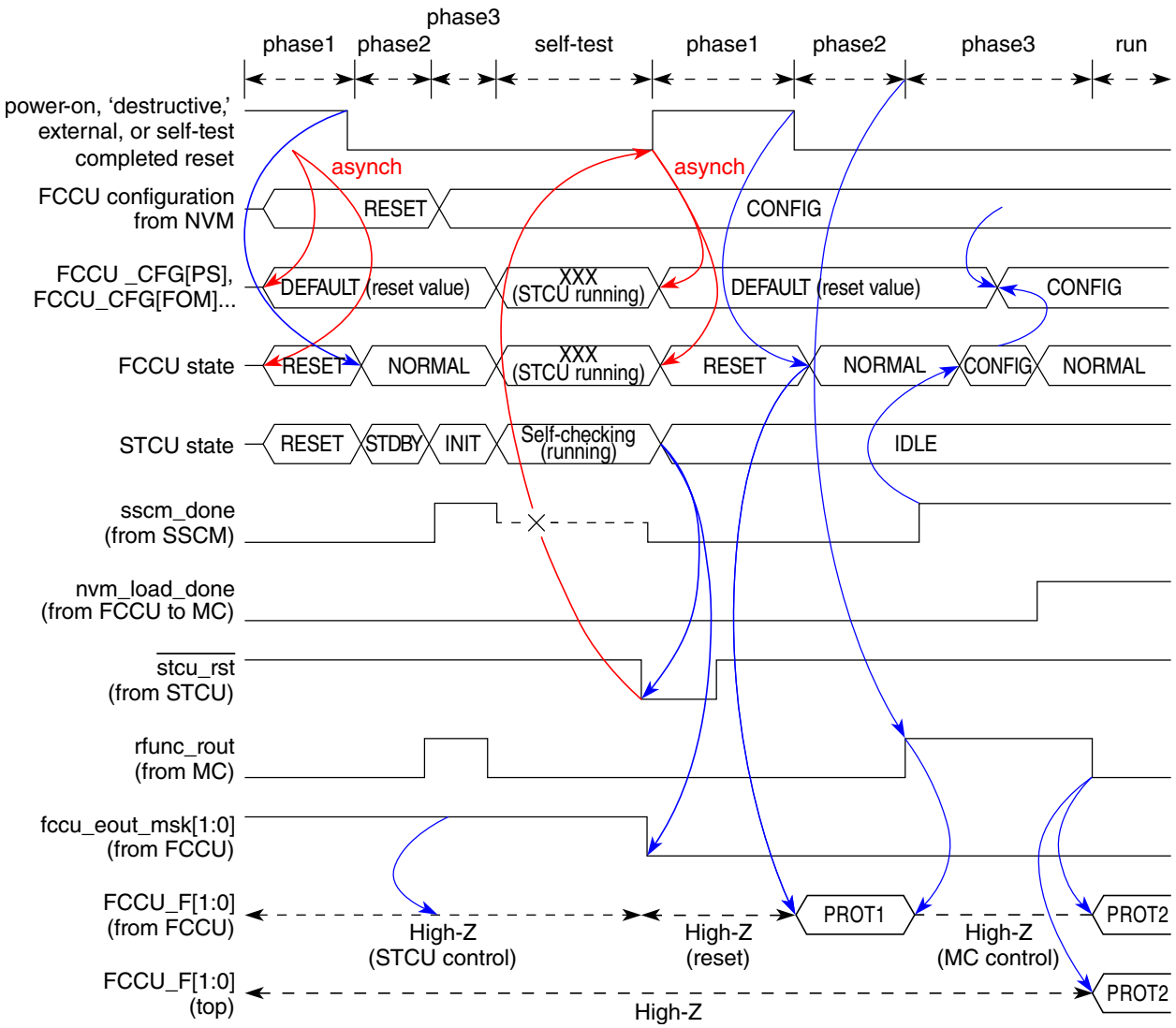


Figure 74-11. NVM interface

74.9.8 EOUT interface

74.9.8.1 Introduction

You use the EOUT[1:0] signals to indicate FCCU's condition to off-chip logic. For information on the availability and names of these FCCU signals on the boundary of this chip, see the chip-specific FCCU information.

74.9.8.2 The FCCU conditions

There are three FCCU conditions:

Condition	Description
Faulty	The fault-output (EOUT) timer is running (see How the fault-output (EOUT) timer works).
Nonfaulty	All of the following are true: • The fault-output (EOUT) timer is not running. • FCCU is in Alarm or Normal state.
Configuration	All of the following are true: • The fault-output (EOUT) timer is not running. • FCCU is in Configuration state.

74.9.8.3 How the fault-output (EOUT) timer works

FCCU starts the fault-output (EOUT) timer when all of the following are true:

- The fault-output (EOUT) signals are not programmed to be always low (FCCU_SET_CLEAR).
- The fault-output (EOUT) timer isn't already running.
- FCCU enters Fault state as the result of a fault.

When the fault-output (EOUT) timer is already running and a new fault occurs:

- When FCCU is in Configuration state: FCCU doesn't restart the fault-output (EOUT) timer.
- When FCCU is in Normal state or Alarm state:
  - And Alarm state is enabled for the fault (noncritical): FCCU enters (or remains in) Alarm state but doesn't restart the fault-output (EOUT) timer.
  - And Alarm state is disabled for the fault (noncritical): FCCU enters Fault state and restarts the fault-output (EOUT) timer.
- When FCCU is in Fault state: FCCU restarts the fault-output (EOUT) timer.

FCCU stops and reinitializes the fault-output (EOUT) timer when all of the following are true:

- If the EOUT signals are in Bistable fault-output mode (FOM), T_min has expired.
- All faults that caused FCCU to enter or remain in Fault state since FCCU started the fault-output (EOUT) timer have been cleared, causing FCCU to return to Normal state.

74.9.8.4 Prepare the EOUT signals to indicate FCCU's condition

Make sure all of the following are true:

- The EOUT signals are active (FCCU_SET_AFTER_RESET).
- The EOUT signals are controlled by FCCU's finite state machine (FCCU_SET_CLEAR).

74.9.8.5 More about the EOUT interface

Because SIUL2 does not control the pad-buffer enable for an Alternate function, FCCU provides a mechanism to drive the pads for EOUT[1:0] in Open-Drain (OD) mode (FCCU_CFG[OD]) so that a pad is in a high-impedance state when FCCU drives a logic 1 and is low when FCCU drives a logic 0.

Different fault-output modes (protocols) for the fault-output (EOUT) interface are supported (FCCU_CFG[FOM]):

- Dual-Rail
- Time-Switching
- Bistable
- Test

You can further configure the fault-output modes using the following attributes:

Attribute	Field	Setting used in the example diagrams and tables that follow
Switching mode	FCCU_CFG[SM]	Slow
Polarity selection	FCCU_CFG[PS]	For the faulty indication, EOUT1 is high, and EOUT0 is low.
Derived prescaler	FCCU_CFG[FOPE], FCCU_CFG[FOP]	0

The external monitor of the EOUT protocol should oversample the EOUT signals in order to synchronize periodically the external clock (used by the monitor) and CLKSAFE (which detects the edge transition of the EOUT protocol) in Dual-Rail or Time-Switching fault-output mode.

In case of a failure event or on software request for EOUT indication, the signal(s) are set to the faulty state for a minimum time T_min, even if software tries to release it before. If software configures the error pins to OK(1), and if a fault comes trying to drive the pin to NOK(0), then priority is given to the fault indication and the error signals indicate NOK,

such as an incoming fault is not masked when software has set the error signal to high. During the T_min by a non-software fault, the FCCU FSM moves independently of this signal state (low), and as soon as the timer expires, the pin behavior is dictated by the state in which the FSM finds itself in, and it is not possible to set the signals to OK by software moving FCCU to Configuration state, as long as this timer is running. No software intervention is needed to bring the signal from the low state.

Software can bring the pin back to OK state by clearing the faults and waiting for the T_min interval to expire, after which the FCCU automatically enters Normal state and the error signal indicates OK.

In case another failure event happens within T_min after a first one, the T_min counter is restarted.

NOTE

The transition from Fault to Configuration state is not possible but is shown to display the behavior in all four states—reset, normal, error, and configuration.

NOTE

FCCU remains in Normal state when a fault is disabled; therefore, *NA* (not applicable) appears in the following table.

NOTE

The following table is valid only when the EOUT signals are active (FCCU_SET_AFTER_RESET).

Table 74-4. FCCU error pin behavior on state transitions

Fault signaling state	Transition from Normal to Fault state			Normal+Configuration states			Transition from Normal to Alarm state		
	with Fault disabled | enabled						with Fault disabled | enabled		
	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad	Internal error pin	Error pin enable	Error pad
Disabled	NA | OK	NA | 1	NA | OK	OK	1	OK	NA | OK	NA | 1	NA | OK
Enabled	NA | NOK	NA | 1	NA | NOK	OK	1	OK	NA | OK	NA | 1	NA | OK

NOTE

The first time FCCU is in Configuration state, the EOUT pads are in a high-impedance state; thereafter, their states reflect the programmed EOUT protocol.

74.9.8.6 Dual-rail protocol

Dual-rail encoding is an alternate method for encoding bits. In contrast with classical encoding, where each signal carries a single-bit value, dual-rail encoded circuits use two wires to carry each bit. The encoding scheme is given in Table 74-5 and the related timing diagram is given in Figure 74-12 and Figure 74-13.

Table 74-5. Dual-rail encoding

Logical state	Dual-rail encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	00	toggling
faulty	11	
reset	Hi impedence[1]	no toggling
configuration	same as NORMAL	toggling

1. Final value depends on the SoC setting at pad level.

In the RESET phase the output pins are set as "high impedance".

Note

Figure 74-12 and Figure 74-13 are formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

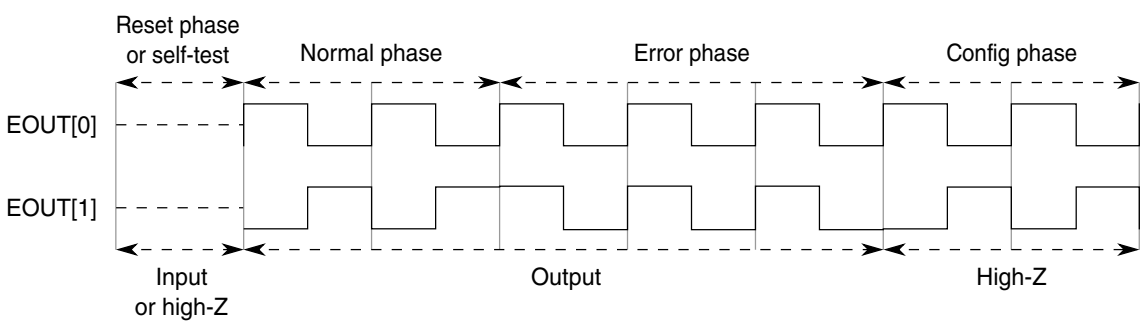


Figure 74-12. Dual-rail protocol (slow switching mode)

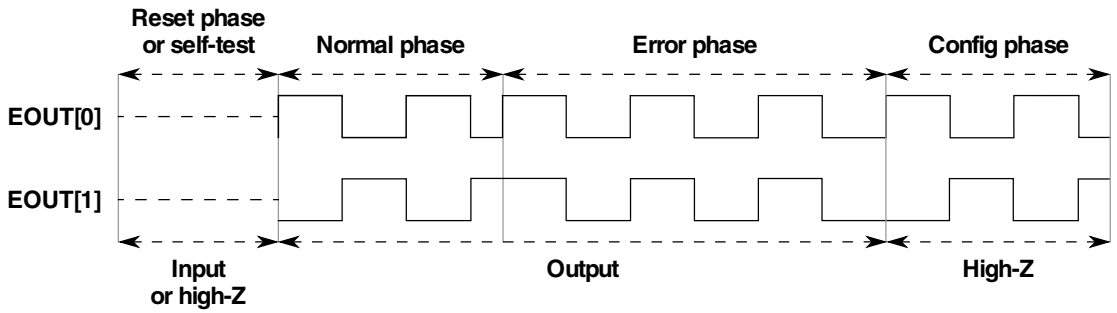


Figure 74-13. Dual-rail protocol (fast switching mode)

74.9.8.7 Time-switching protocol

The encoding scheme is given in Table 74-6 and the related timing diagram is given in Figure 74-14.

Table 74-6. Time-switching encoding

Logical state	Time-switching encoding (output pins EOUT[1:0])	Note
non-faulty	10	toggling
non-faulty	01	
faulty	10	no toggling
reset	high-Z[1]	no toggling
configuration	same as NORMAL	toggling

1. Final value depends on device specific settings at pad level.

As long as FCCU is in NORMAL or ALARM state, outputs will show "non-faulty" signal. Output pins #0, #1 will toggle between 01 and 10 with frequency specified by the CFG[FOP].

In the FAULT state, the output pin EOUT[0] is set as low.

In Time Switching mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".[2]

Note

Figure 74-14 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

2. Actual value depends on device specific settings at pad level.

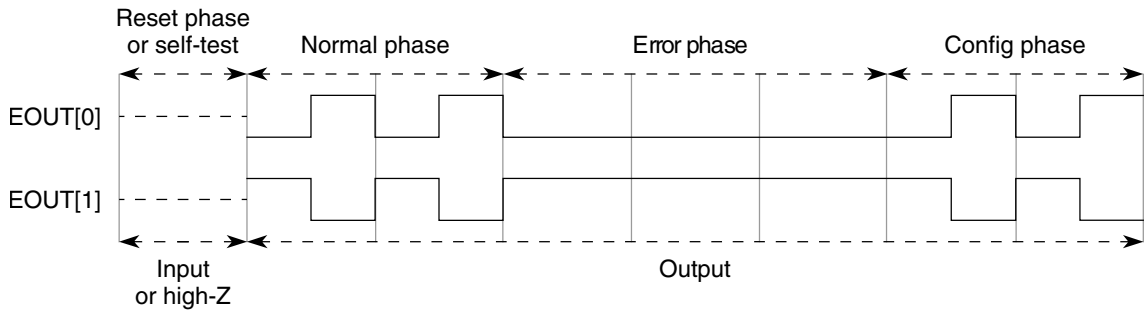


Figure 74-14. Time-switching protocol

74.9.8.8 Bi-stable protocol

The encoding scheme is given in Table 74-7 and the related timing diagram is given in Figure 74-15.

Table 74-7. Bi-stable encoding

Logical state	Bi-stable encoding (output pins EOUT[1:0])	Note
non-faulty	01	no toggling
faulty	10	no toggling
reset	high-Z[1]	no toggling
configuration	same as NORMAL	no toggling

1. Final value depends on device specific settings at pad level.

In the FAULT state, the faulty logical state is indicated. In NORMAL or ALARM state, "no-faulty" state is indicated. In Bi-stable mode the second output (EOUT[1]) is the inverted signal of first output (EOUT[0]).

In the RESET phase the output pins are set as "high impedance".[3]

Note

Figure 74-15 is formatted to display the behavior in all four phases (reset, normal, error, and config), not to imply transitions between one phase to another. In particular, transition from error phase to config phase is not possible.

3. Actual value depends on device specific settings at pad level.

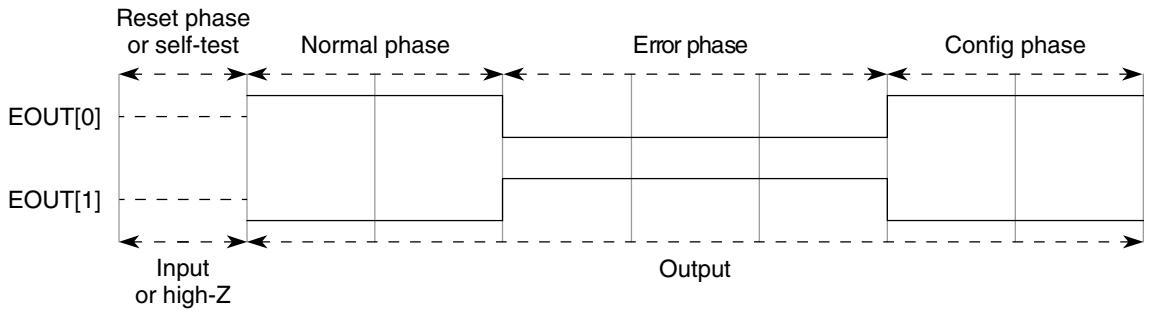


Figure 74-15. Bi-stable protocol

74.9.8.9 Option 1, Option 2

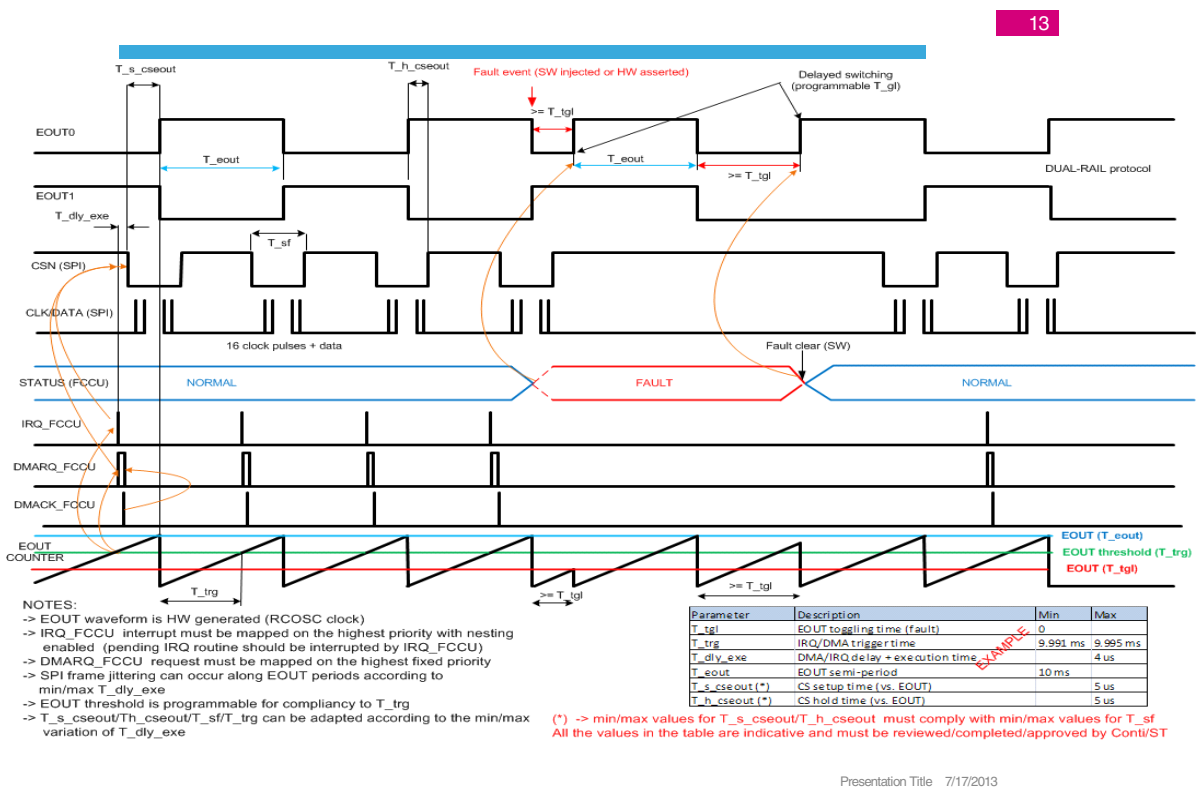


Figure 74-16. Option1 , Option 2

The internal counter used to generate the EOUT waveforms is used to control a IRQ or DMA trigger event. This event shall appear only when the FCCU is not in FAULT mode as showin in the above figure.

NOTE

While switching fom Non-Fault to Fault state or vice versa, Delay equal to T_Tgl will only be visible if EOUT COUNTER is less than T_Tgl.

74.9.8.10 Option 3

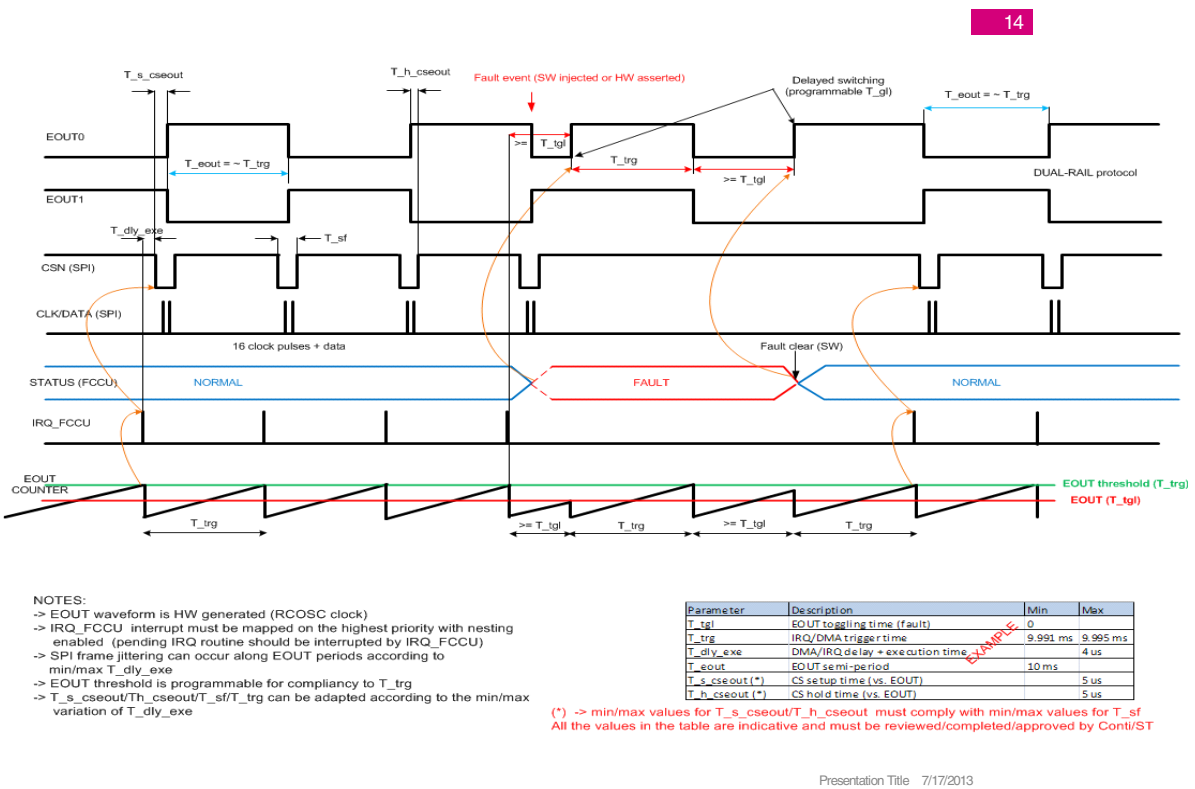


Figure 74-17. Option3

When FCCU_CFG. FOME = 11, the EOUT pin state is controlled by the application SW as far as the FCCU is in NORMAL or ALARM state. As soon as a the FCCU enters the FAULT state, both the EOUT pins are forced by the HW to the same level, which can be 0-0 or 1-1 depending on the polarity configuration. As far as the FCCU is in FAULT

state, the SW has no control over the EOUT pins. Once the fault condition is recovered, the FCCU returns to the NORMAL state and the SW can take back the control of the EOUT pins. Figure 24 shows how this is accomplished. As far as the FCCU is not in FAULT mode, the IRQ interrupts are asserted when the FCCU internal timer reaches the programmed threshold. If needed, it is responsibility of the SW to toggle the EOUT pins within the ISR. The EOUT pin toggling period depends on the IRQ assertion frequency (controlled by the HW) plus the interrupt service time and SW execution. This period will be stable and close to the IRQ programmed period as far as the interrupt service routine can be serviced in a deterministic time frame and executed at the highest priority. During the ISR, additional operations can be done: the figure shows an example of a DSPI frame sent before every EOUT toggling, but with the DSPI CS being kept asserted for some time after the EOUT has toggled. When the FCCU enters into the FAULT state, no IRQ interrupt is fired anymore and the EOUT pins are controlled by the HW. During this time frame, the EOUT pin toggling is exactly matching the programmed period.

74.9.9 Error signal flow

See the chip-specific FCCU information for the diagram of the signal flow.

74.10 Register descriptions

The FCCU registers are listed in the table below. Any address offset not explicitly mentioned in this table is reserved.

The FCCU supports word (32-bit), half-word (16-bit), and byte (8-bit) read and write accesses.

Follow these register-access guidelines:

- Don’t read from or write to any addresses that are not shown in the following table. Doing so may result in a transfer error.
- Don’t write to any of the configuration registers unless FCCU is in Configuration state. Doing so results in a transfer error.
- Don’t write to FCCU_TRANS_LOCK or FCCU_PERMNT_LOCK unless your code runs in Supervisor mode. Doing so results in a transfer error.

All the registers accessible in write mode only in CONFIG state are referred as configuration registers. These configuration registers return to the default value after configuration watchdog timer expires. These are the registers protected by FCCU_TRANS_LOCK and FCCU_PERMNT_LOCK registers. The configuration register setting has effect only when the FCCU state exits from the CONFIG state.

For each possible NCF failure source, a different reaction—including no reaction—is configurable through the use of NMI, IRQ, and long/short reset selection registers. It is not possible for a single event upset to switch off all reactions on failures as implementation is per fault source (but it will be possible to switch them all off by SW if intended). Failures themselves are not able to disable all reactions and indications.

The FCCU is not reset by short or long functional resets.

FCCU memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Control (FCCU_CTRL)	32	R/W	0000_00C0h	74.10.1/ 3164
4	Control Key (FCCU_CTRLK)	32	W	0000_0000h	74.10.2/ 3166
8	Configuration (FCCU_CFG)	32	R/W	See section	74.10.3/ 3167
1C	Noncritical Fault Configuration (FCCU_NCF_CFG0)	32	R/W	See section	74.10.4/ 3171
20	Noncritical Fault Configuration (FCCU_NCF_CFG1)	32	R/W	See section	74.10.4/ 3171
24	Noncritical Fault Configuration (FCCU_NCF_CFG2)	32	R/W	See section	74.10.4/ 3171
4C	Noncritical Fault State Configuration (FCCU_NCFS_CFG0)	32	R/W	See section	74.10.5/ 3172
50	Noncritical Fault State Configuration (FCCU_NCFS_CFG1)	32	R/W	See section	74.10.5/ 3172
54	Noncritical Fault State Configuration (FCCU_NCFS_CFG2)	32	R/W	See section	74.10.5/ 3172
58	Noncritical Fault State Configuration (FCCU_NCFS_CFG3)	32	R/W	See section	74.10.5/ 3172
5C	Noncritical Fault State Configuration (FCCU_NCFS_CFG4)	32	R/W	See section	74.10.5/ 3172
80	Noncritical Fault Status (FCCU_NCF_S0)	32	R/W	0000_0000h	74.10.6/ 3173
84	Noncritical Fault Status (FCCU_NCF_S1)	32	R/W	0000_0000h	74.10.6/ 3173
88	Noncritical Fault Status (FCCU_NCF_S2)	32	R/W	0000_0000h	74.10.6/ 3173
90	Noncritical Fault Key (FCCU_NCFK)	32	W	0000_0000h	74.10.7/ 3175

Table continues on the next page...

FCCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
94	Noncritical Fault Enable (FCCU_NCF_E0)	32	R/W	See section	74.10.8/ 3175
98	Noncritical Fault Enable (FCCU_NCF_E1)	32	R/W	See section	74.10.8/ 3175
9C	Noncritical Fault Enable (FCCU_NCF_E2)	32	R/W	See section	74.10.8/ 3175
A4	Noncritical Fault Timeout Enable (FCCU_NCF_TOE0)	32	R/W	See section	74.10.9/ 3176
A8	Noncritical Fault Timeout Enable (FCCU_NCF_TOE1)	32	R/W	See section	74.10.9/ 3176
AC	Noncritical Fault Timeout Enable (FCCU_NCF_TOE2)	32	R/W	See section	74.10.9/ 3176
B4	Noncritical Fault Timeout (FCCU_NCF_TO)	32	R/W	See section	74.10.10/ 3177
B8	Configuration-State Timer Interval (FCCU_CFG_TO)	32	R/W	See section	74.10.11/ 3178
BC	IO Control (FCCU_EINOUT)	32	R/W	See section	74.10.12/ 3179
C0	Status (FCCU_STAT)	32	R	0000_0010h	74.10.13/ 3180
C4	NA Freeze Status (FCCU_N2AF_STATUS)	32	R	0000_0000h	74.10.14/ 3182
C8	AF Freeze Status (FCCU_A2FF_STATUS)	32	R	0000_0000h	74.10.15/ 3183
CC	NF Freeze Status (FCCU_N2FF_STATUS)	32	R	0000_0000h	74.10.16/ 3184
D0	FA Freeze Status (FCCU_F2A_STATUS)	32	R	0000_0000h	74.10.17/ 3185
DC	Noncritical Fault Fake (FCCU_NCFF)	32	R/W	0000_0000h	74.10.18/ 3186
E0	IRQ Status (FCCU_IRQ_STAT)	32	R/W	0000_0000h	74.10.19/ 3187
E4	IRQ Enable (FCCU_IRQ_EN)	32	R/W	0000_0000h	74.10.20/ 3189
E8	XTMR (FCCU_XTMR)	32	R	0000_0000h	74.10.21/ 3190
EC	Mode Controller Status (FCCU_MCS)	32	R	See section	74.10.22/ 3191
F0	Transient Configuration Lock (FCCU_TRANS_LOCK)	32	W	0000_0000h	74.10.23/ 3193
F4	Permanent Configuration Lock (FCCU_PERMNT_LOCK)	32	W	0000_0000h	74.10.24/ 3194
F8	Delta T (FCCU_DELTA_T)	32	R/W	270F_0000h	74.10.25/ 3194

Table continues on the next page...

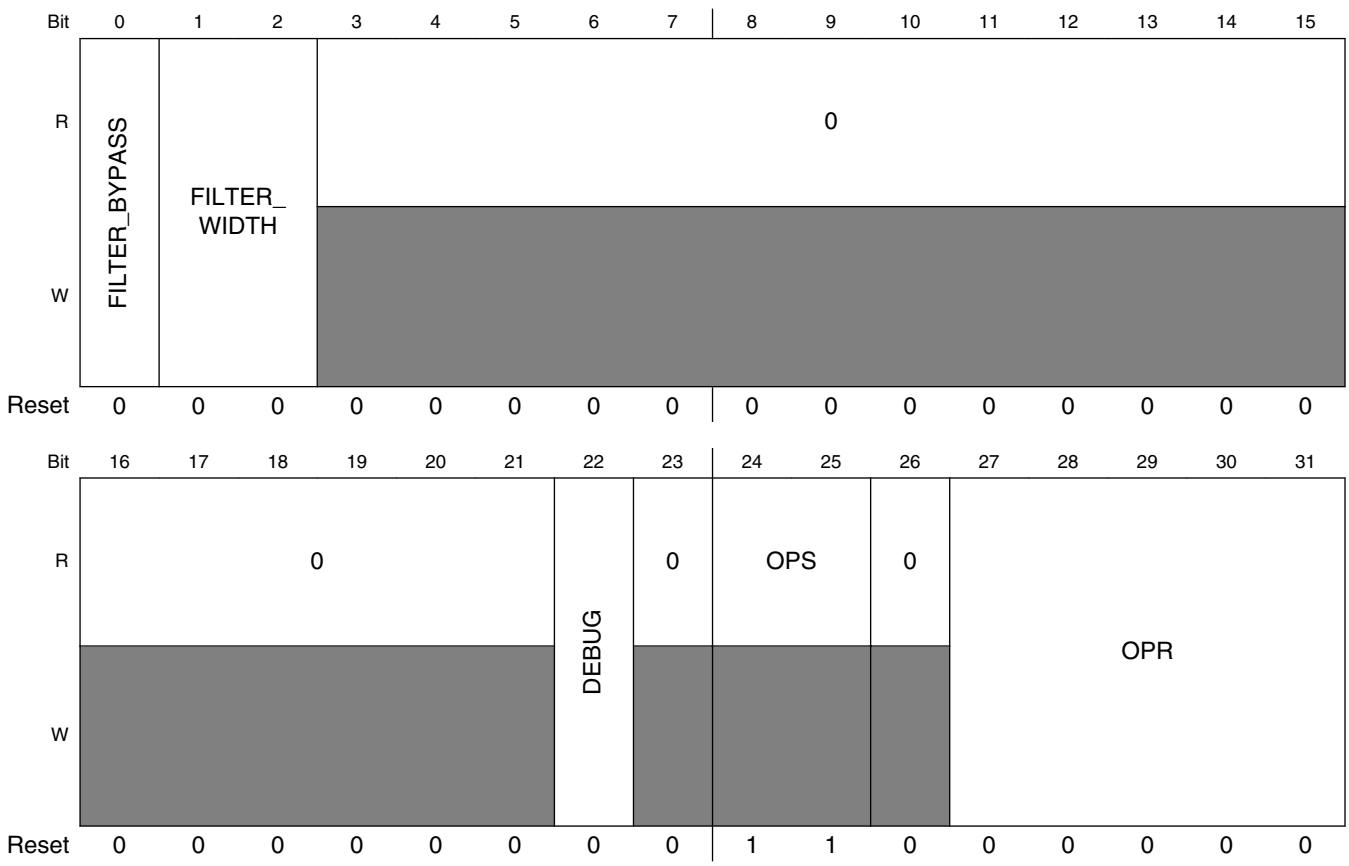
FCCU memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
FC	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN0)	32	R/W	0000_0000h	74.10.26/ 3195
100	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN1)	32	R/W	0000_0000h	74.10.26/ 3195
104	IRQ Alarm Enable (FCCU_IRQ_ALARM_EN2)	32	R/W	0000_0000h	74.10.26/ 3195
10C	NMI Enable (FCCU_NMI_EN0)	32	R/W	0000_0000h	74.10.27/ 3196
110	NMI Enable (FCCU_NMI_EN1)	32	R/W	0000_0000h	74.10.27/ 3196
114	NMI Enable (FCCU_NMI_EN2)	32	R/W	0000_0000h	74.10.27/ 3196
11C	Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN0)	32	R/W	0000_0000h	74.10.28/ 3197
120	Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN1)	32	R/W	0000_0000h	74.10.28/ 3197
124	Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN2)	32	R/W	0000_0000h	74.10.28/ 3197

74.10.1 Control (FCCU_CTRL)

Initiates and indicates the status of operations, indicates that FCCU successfully loaded configuration data from the NVMCFG signals, controls glitch-filter settings, and enables Debug mode.

Address: 0h base + 0h offset = 0h



FCCU_CTRL field descriptions

Field	Description
0 FILTER_ BYPASS	Filter bypass 0 glitch filter not bypassed 1 glitch filter bypassed
1–2 FILTER_WIDTH	Filter width 00 filters glitches up to 50 µs 01 filters glitches up to 75 µs 10 filters glitches up to 100 µs 11 filters glitches up to 100 µs

Table continues on the next page...

FCCU_CTRL field descriptions (continued)

Field	Description
3–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 DEBUG	Debug Mode Enable Specifies whether Debug mode is enabled. If so, FCCU enters Debug mode when the Debug signal is asserted. When FCCU enters Debug mode, it halts operation and remains in the state it was in before it entered Debug mode. 0 Disabled 1 Enabled
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 OPS	Operation status. This bit can be read and cleared (via OP15 operation) by the software. 00 Idle 01 In progress 10 Aborted 11 Successful
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 OPR	Operation Run Initiates operations that perform actions such as putting FCCU in Configuration state or reading a timer. For information on how to run operations, see Run operations. FCCU ignores any write to this field while the operation status (OPS) is In Progress. After completion of an operation, FCCU sets this field to OP0. The following events result in an operation status (OPS) of Aborted: • Writing to a Noncritical Fault Status (FCCU_NCF_S*n*) register (which automatically initiates the OP12 operation) without first successfully unlocking the register (NCFK) • Initiating an OP1 operation when FCCU is not in Normal state or the configuration is locked • Initiating an OP1, OP2, or OP31 operation without first unlocking the operation (CTRLK) 00000 OP0—No operation 00001 OP1—Applies only when the configuration is unlocked, when FCCU is in Normal state, and immediately after you unlock the operation (CTRLK). Put FCCU in Configuration state. 00010 OP2—Applies only immediately after you unlock the operation (CTRLK). Put FCCU in Normal state. 00011 OP3—Capture FCCU's state and the states of the EOUT signals in the Status (FCCU_STAT) register. 00100 OP4—Read the FCCU frozen status flags (NA Freeze Status (FCCU_N2AF_STATUS)). 00101 OP5—Read the FCCU frozen status flags (AF Freeze Status (FCCU_A2FF_STATUS)). 00110 OP6—Read the FCCU frozen status flags (NF Freeze Status (FCCU_N2FF_STATUS)). 00111 OP7—Read the FCCU frozen status flags (FA Freeze Status (FCCU_F2A_STATUS)). 01000 Reserved 01001 Reserved 01010 OP10—Read the Noncritical Fault Status (FCCU_NCF_S*n*) register. 01011 Reserved

Table continues on the next page...

FCCU_CTRL field descriptions (continued)

Field	Description
	01100 OP12—Do not initiate this operation; it is automatically initiated by FCCU. A Noncritical Fault Status (FCCU_NCF_S*n*) register status clear operation is in progress. 01101 OP13—Clear the freeze status registers (see the freeze registers). 01110 OP14—Do not initiate this operation; it is automatically initiated by FCCU. A Configuration-state timeout is in progress. For more information, see Configuration registers. 01111 OP15—Set the operation status (OPS) to Idle. 10000 Reserved 10001 OP17—Read the Alarm-state timer (XTMR). 10010 Reserved 10011 OP19—Read the Configuration-state timer (XTMR). 10100 OP20—Read the Error Pin low counter value (XTMR). 10101–11110 Forbidden. Writing any of these values returns an operation status (OPS) of Aborted with no side effect. 11111 OP31—Applies only immediately after you unlock the operation (CTRLK). Load configuration data from the NVMCFG signals to certain fields in the Configuration (FCCU_CFG) register, but without resetting FCCU (for testing and debugging). See FCCU_CFG register bit value sources (N and C) by event. The operation status (OPS) for this operation is always Successful.

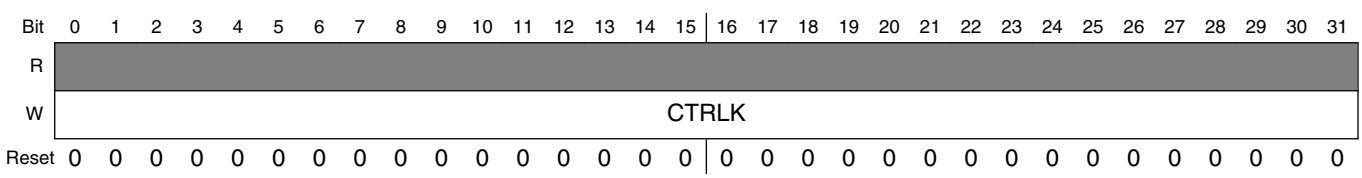
74.10.2 Control Key (FCCU_CTRLK)

Writable only with a 32-bit write. Unlocks locked operations so you can initiate them (OPR). For information on how to unlock locked operations before you initiate them, see Run an operation.

NOTE

Reading from this register always returns the value 0000_0000h.

Address: 0h base + 4h offset = 4h



FCCU_CTRLK field descriptions

Field	Description
0–31 CTRLK	Locked-Operation Control Key Unlocks locked operations so you can initiate them (OPR). For information on how to unlock locked operations before you initiate them, see Run an operation.

FCCU_CTRLK field descriptions (continued)

Field	Description
	NOTE: You must initiate an operation in the instruction that immediately follows the instruction that unlocks it; otherwise, the operation is again locked. 9137_56AFh: Unlock OP1. 825A_132Bh: Unlock OP2. 29AF_8752h: Unlock OP31. Any other value: Do nothing.

74.10.3 Configuration (FCCU_CFG)

Writable only when FCCU is in Configuration state. Specifies the global configuration for FCCU.

The values of the bits in this register are affected by writes as well as any of the following events:

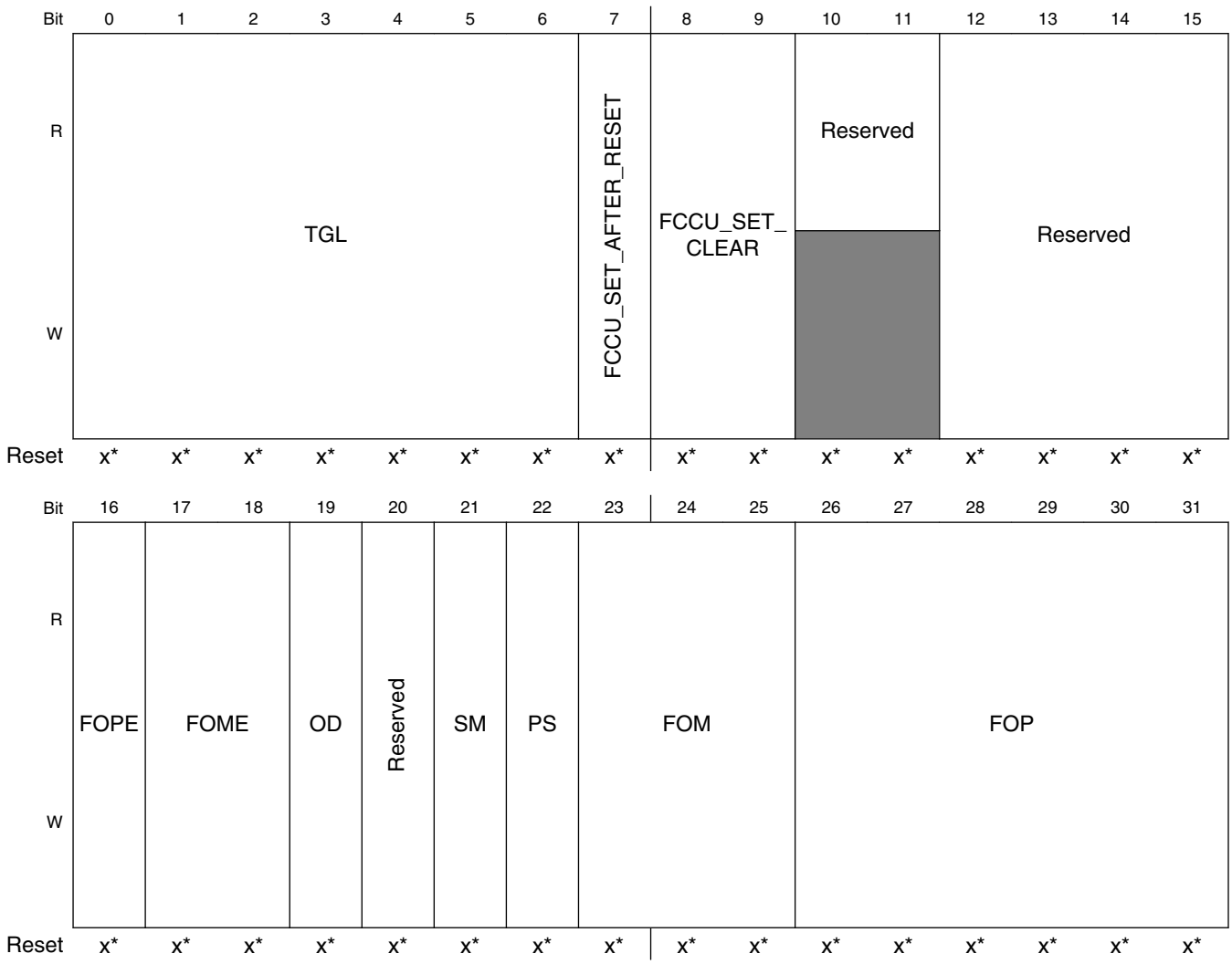
- The chip resets FCCU.
- FCCU automatically leaves Configuration state and enters Normal state because the configuration-timer interval (TO) expires. This is called a Configuration-state timeout.
- You run an OP31 operation.

For information on the sources of bit values for these events, see FCCU_CFG register bit value sources (N and C) by event.

NOTE

If you specify a new value for any of the fields in this register that affect the EOUT signals while the fault-output (EOUT) timer is running (FCCU is indicating a fault on the EOUT signals), FCCU doesn't use the new settings you specified until after the fault-output (EOUT) timer expires (FCCU stops indicating a fault on the EOUT signals).

Address: 0h base + 8h offset = 8h



* Notes:

- For information on this register's reset value, see FCCU_CFG register bit value sources (N and C) by event.x = Undefined at reset.

FCCU_CFG field descriptions

Field	Description
0–6 TGL	EOUT Toggling Time Active only if CFG[SM] specifies slow switching. This value is the minimum duration of the EOUT pulse during Fault -> Normal/Alarm and Normal/Alarm -> Fault state transitions. For more information, see the FOME field description for the setting 11b. Min = 1 µs, Max = 128 µs, resolution = 1 µs
7 FCCU_SET_AFTER_RESET	Fault-Output (EOUT) Activate For fault-output (EOUT) signaling, controls whether the EOUT signals are active.

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
	0 Inactive (the EOUT signals are in a high-impedance state) 1 Active (the EOUT signals indicate FCCU's condition)
8–9 FCCU_SET_CLEAR	Fault-Output (EOUT) Control Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Controls whether the fault-output (EOUT) signals are controlled by FCCU's finite state machine (FSM). 00 Controlled by the FSM 01 Always low 10 Controlled by the FSM 11 High until a fault occurs on a channel, regardless of whether that fault is disabled; thereafter, controlled by the FSM. Note: FCCU ignores an attempt to write this value if the fault-output (EOUT) timer is already running.
10–11 Reserved	This field is reserved.
12–15 Reserved	This field is reserved. Always write the reset value to this field.
16 FOPE	Fault-Output (EOUT) Prescaler Extension For fault-output (EOUT) signaling, specifies the most significant bit of the derived prescaler that controls the signaling frequency. For more information, see the description of the FOP field.
17–18 FOME	Fault Output Mode Extension 00 Current behavior (default) 01 Option #1: HW interrupt trigger(IRQ_EOUT) asserted by the FCCU when EOUT reaches T_Trig value when in NORMAL/ALARM state. (EOUT counter saturation is shown in Option 1, Option 2). 10 Option #2: DMA request (DMARQ_EOUT) asserted by the FCCU when EOUT reaches T_Trig value (EOUT counter saturation as shown in Option 1, Option 2) when in NORMAL/ALARM state. 11 Option #3 only if CFG[FOM] is 000b: EOUT protocol is SW controlled in NORMAL/ALARM state and HW controlled in FAULT state (dual rail, in phase toggling with programmable delay) following a HW interrupt trigger (IRQ_EOUT) asserted by the FCCU when EOUT reaches T_Trig value (EOUT counter saturation as shown in Option 3 section) when in NORMAL/ALARM state in order to generate the EOUT protocol via SW. An alternative option is to use an external eTMR. EOUT dual-rail protocol: • NORMAL/ALARM: for example, 10 ms toggling in opposite phase • FAULT: for example, 10 ms toggling in phase starting after a programmable delay T_tgl, slow switching mode (SM is 0) • FAULT: fast switching mode (SM is 1) in phase (no periodic toggling). For details, please refer to Option 1, Option 2 section. For details, please refer to Option 3 section. If option #3 is disabled when CFG[FOM] is 000b, the result is dual-rail mode (current behavior).
19 OD	Open-Drain Mechanism to select between Push-pull and Open drain(OD) mode for the error indicating pin(s) 0 Push-pull 1 Open-drain
20 Reserved	This field is reserved. Always write the reset value to this field.
21 SM	Fault-Output (EOUT) Switching Mode

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
	For fault-output (EOUT) signaling, controls how quickly the signals switch between faulty and nonfaulty indication. Applies only to Dual-Rail and Time-Switching fault-output modes. 0 Slow: No EOUT frequency violation during the FCCU state transition (Normal to Fault or vice versa, and Configuration to Normal). The indication transition occurs after a maximum delay equal to the duration of the semiperiod of the EOUT frequency. 1 Fast: The indication transition (Normal to Fault or vice versa, and Configuration to Normal) occurs immediately. A pulse with the minimum duration corresponding to the CLKSAFE period can occur in this mode. It implies a frequency violation of the EOUT protocol.
22 PS	Fault-Output (EOUT) Polarity Selection For fault-output (EOUT) signaling, controls the polarity of the signals for fault-output-mode indications that hold the signals low or high (versus toggling them or placing them in a high-impedance state). Applies only to Time-Switching fault-output mode (for faulty and configuration indications) and Bistable fault-output mode (for all indications). 0 For the faulty indication, EOUT1 is high, and EOUT0 is low. 1 For the faulty indication, EOUT1 is low, and EOUT0 is high.
23–25 FOM	Fault-Output (EOUT) Mode For fault-output (EOUT) signaling, controls the protocol of the signaling. **NOTE:** In the test modes, a simple double-stage resynchronization stage is used to resynchronize the EOUT input/outputs with CLKSAFE. 000 Dual-Rail (default state) [EOUT[1:0]= outputs] 001 Time-Switching [EOUT[1:0]= output to be used] 010 Bistable 011 Reserved 100 Reserved 101 Test 0 (controlled by the FCCU_EINOUT register; EOUT1 is an output; EOUT0 is an input) 110 Test 1 (controlled by the FCCU_EINOUT register; EOUT1 and EOUT0 are both outputs) 111 Test 2 (controlled by the FCCU_EINOUT register; EOUT1 is an input; EOUT0 is an output)
26–31 FOP	Fault-Output (EOUT) Prescaler For fault-output (EOUT) signaling, specifies the least significant bits of the derived prescaler that controls the signaling frequency. The derived prescaler is the concatenation of FOPE and FOP and is represented by *{FOPE, FOP}*. The following equation defines the signaling frequency in terms of the CLKSAFE frequency ($CLKSAFE_{freq}$) and the derived prescaler: $EOUT_{freq} = CLKSAFE_{freq} / ((\{FOPE, FOP\} + 1) \times 2 \times 2048)$ The following example values and meanings are for the derived (7-bit) prescaler, not FOP alone. The meanings show the derived prescaler followed by the corresponding value by which the input clock frequency is then divided. 0000000: 0; divide by 2 × 2048 (4096) 0000001: 1; divide by 4 × 2048 (8192) 0000010: 2; divide by 6 × 2048 (12,288) … 0001000: 8; divide by 18 × 2048 (36,864)

Table continues on the next page...

FCCU_CFG field descriptions (continued)

Field	Description
	… 1111101: 125; divide by 252 × 2048 (516,096) 1111110: 126; divide by 254 × 2048 (520,192) 1111111: 127; divide by 256 × 2048 (524,288)

74.10.4 Noncritical Fault Configuration (FCCU_NCF_CFG*n*)

Contains the configuration of noncritical faults in terms of fault-recovery management. The configuration depends on the type of signaling of a fault event. Hardware-recoverable faults should be configured only if a previous latching stage captures and holds the physical fault; otherwise, the fault can be lost. All other faults should be configured as software faults.

NOTE

These registers are writable only when FCCU is in Configuration state.

Noncritical Fault Configuration (FCCU_NCF_CFG*n*) register (value of n)	Offset	NCFCx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	1Ch	31	0
1	20h	63	32
2	24h	95	64

Address: 0h base + 1Ch offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:

- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_CFG*n* field descriptions

Field	Description
0–31 NCFCx	Noncritical Fault Configuration. Defines the fault recovery mode. Hardware-recoverable faults are self-recovered (status flag clearing and related) if the root cause (input fault) has been removed. Software-recoverable faults are recovered (status flag clearing) by software clearing of the related status flag.

FCCU_NCF_CFG*n* field descriptions (continued)

Field	Description
	For the mapping of the NCFCx fields among registers, see the earlier table in this register description. 0 Hardware-recoverable fault 1 Software-recoverable fault

74.10.5 Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*)

The FCCU_NCFS_CFGx registers contain the configuration of each noncritical fault in terms of fault reaction (short or long functional reset request pulse) when it is the root cause for the FAULT state transition.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

Table 74-8 shows FCCU configuration register channels.

Table 74-8. FCCU NCFS configuration register channels

Address offset	Register name	Channel range (x)
4Ch	FCCU_NCFS_CFG0	NCFSCx[15:0]
50h	FCCU_NCFS_CFG1	NCFSCx[31:16]
54h	FCCU_NCFS_CFG2	NCFSCx[47:32]
58h	FCCU_NCFS_CFG3	NCFSCx[63:48]
5Ch	FCCU_NCFS_CFG4	NCFSCx[79:64]

Address: 0h base + 4Ch offset + (4d × i), where i=0d to 4d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFSCx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCFS_CFG*n* field descriptions

Field	Description
0–31 NCFSCx	Noncritical fault state configuration See Table 74-8 for register offset to channel number relationship. 00 No reset reaction 01 Short functional reset request pulse (FAULT state reaction)

FCCU_NCFS_CFG*n* field descriptions (continued)

Field	Description
	10 Long functional reset request pulse (FAULT state reaction) 11 No reset reaction

74.10.6 Noncritical Fault Status (FCCU_NCF_S*n*)

The FCCU_NCF_Sx register contains the latched fault indication collected from the noncritical fault sources. Faults are latched also in the CONFIG state and independently from the enabling or reactions programmed for the NCF.

No reactions are executed until the FCCU moves in the NORMAL state.

FCCU reacts and moves from the NORMAL state into the ALARM state only if the respective enable bit for a fault is set in the FCCU_NCF_Ex register and the respective enable bit for the timeout is set in the FCCU_TOEx register.

FCCU reacts and moves from the NORMAL or ALARM state into the FAULT state if the respective enable bit for a fault is set in the FCCU_NCF_Ex register and the respective enable bit for the timeout is disabled in the FCCU_TOEx register.

FCCU reacts and moves from the ALARM state into the FAULT state if the timeout (FCCU_TO register) is elapsed before recovering from the fault.

The timeout is stopped only when the FCCU returns in the NORMAL state.

The FCCU_NCF_Sx register is encoded respectively into the N2FF_STATUS or A2FF_STATUS register to freeze the entry condition in the FAULT state. The status bits of the FCCU_NCF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the proper key into the FCCU_NCFK register.
2. Clear the status (flag) bit NCFSx => the opcode OP12 is automatically set into the FCCU_CTRL.OPR field.
3. Wait for the completion of the operation (FCCU_CTRL.OPS field).
4. Read the FCCU_NCF_Sx register in order to verify the effective deletion and in case of failure to repeat the sequence.

As result of the above sequence, in addition the FAULT interface provides support to clear the external fake FAULT root .

NOTE

There should not be any other operation in between the above steps.

The FCCU moves from the FAULT or ALARM state into the NORMAL state if all the source faults that caused the transition into the FAULT state have been removed (HW recoverable fault) or cleared via SW (SW recoverable fault). In case of nested faults that are not all recovered, the FCCU will remain in the FAULT or ALARM state.

The SW application executes the FCCU_NCF_Sx read operation by the following sequence:

1. Set the OP10 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_NCF_Sx register.

In case of re-configuration of the FCCU (CONFIG state), before to return in NORMAL state the pending status bits into the FCCU_NCF_Sx must be cleared in order to avoid a false transition in ALARM/FAULT state.

The following faults are ignored:

- to write a wrong key into the FCCU_NCFK register
- to attempt to clear a HW recoverable fault

Noncritical Fault Status (FCCU_NCF_S*n*) register (value of n)	Offset	NCFSx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	80h	31	0
1	84h	63	32
2	88h	95	64

Address: 0h base + 80h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	NCFSx																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCF_S*n* field descriptions

Field	Description
0–31 NCFSx	Noncritical fault status. The status bits related to the noncritical fault configured as HW recoverable faults are read-only and the flag is self cleared when the fault source is removed. For the mapping of the NCFSx fields among registers, see the earlier table in this register description. 0 No "noncritical" fault latched 1 "Noncritical" fault latched

74.10.7 Noncritical Fault Key (FCCU_NCFK)

The FCCU_NCFK register implements the key access to clear the status flags of the FCCU_NCF_Sx register.

The status bits of the FCCU_NCF_Sx register, configured as SW recoverable faults, can be cleared by the following locked sequence:

1. Write the key into the FCCU_NCFK register.
2. Clear the status (flag) bit NCFSx .

The key must be written for each FCCU_NCF_Sx clear operation.

The FCCU_NCFK register is not readable; a 0000_0000h value is always returned in case of read operation.

Address: 0h base + 90h offset = 90h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W	NCFK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCFK field descriptions

Field	Description
0–31 NCFK	Noncritical fault key = AB34_98FEh

74.10.8 Noncritical Fault Enable (FCCU_NCF_E*n*)

The FCCU_NCF_En registers enable the fault sources to allow a transition from the NORMAL into the FAULT or ALARM state. In case of fault masking, the respective status bit into the FCCU_NCF_Sn register is set (for debugging purposes), only the reaction is masked.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

NOTE

Any enabled fault should be programmed to result in a defined action. For example, set up an ALARM IRQ action by

programming the IRQ Alarm Enable Register (FCCU_IRQ_ALARM_ENn).

Noncritical Fault Enable (FCCU_NCF_E*n*) register (value of n)	Offset	NCFEx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	94h	31	0
1	98h	63	32
2	9Ch	95	64

Address: 0h base + 94h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFEx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_E*n* field descriptions

Field	Description
0–31 NCFEx	Noncritical fault enable For the mapping of the NCFEx fields among registers, see the earlier table in this register description. 0 No actions following the respective noncritical fault assertion 1 FCCU moves to ALARM or FAULT state

74.10.9 Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*)

The FCCU_NCF_TOEx registers enable a transition from the NORMAL state into the ALARM state if the respective noncritical fault is enabled (NCFEx and NCFTOEx are set). In case the respective timeout is disabled (NCFTOEx is cleared) and the noncritical fault is enabled (NCFEx is set) the FCCU moves into the FAULT state if the related noncritical fault is asserted. The timer (preset with the timeout value defined by FCCU_TO register) is started when the FCCU moves into the ALARM state. If the fault is not recovered within the timeout the FCCU moves from the ALARM state to the FAULT state.

NOTE

These registers are writable only when the FCCU is in the CONFIG state.

Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*) register (value of n)	Offset	NCFTOEx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	A4h	31	0
1	A8h	63	32
2	ACh	95	64

Address: 0h base + A4h offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NCFTOEx																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_TOE*n* field descriptions

Field	Description
0–31 NCFTOEx	Fault timeout enable For the mapping of the NCFTOEx fields among registers, see the earlier table in this register description. 0 FCCU moves into the FAULT state if the respective fault is enabled 1 FCCU moves into the ALARM state if the respective fault is enabled

74.10.10 Noncritical Fault Timeout (FCCU_NCF_TO)

Defines the preset value of the timer for the recovery of enabled noncritical faults. Once FCCU enters Alarm state, following the assertion of a noncritical fault that is enabled (NCFEa and NCFTOEa are set), the timer starts the countdown.

If the fault is not recovered within the timeout period, the FCCU moves from the Alarm state to the Fault state.

NOTE

This register is writable only when FCCU is in Configuration state.

Address: 0h base + B4h offset = B4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	TO																															
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_NCF_TO field descriptions

Field	Description
0–31 TO	Noncritical Fault Timeout Timeout = (TO) × $T_{RC16MHz}$

74.10.11 Configuration-State Timer Interval (FCCU_CFG_TO)

Writable only when FCCU is in Normal, Alarm, or Fault state (not in Configuration state). Not accessible while a Configuration-state timeout (OP14 operation) is in progress. Controls the maximum amount of time that FCCU can be in Configuration state.

NOTE

When a Configuration-state timeout occurs, FCCU changes the value of this register to its reset value.

Address: 0h base + B8h offset = B8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																													TO		
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information.x = Undefined at reset.

FCCU_CFG_TO field descriptions

Field	Description
0–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 TO	Configuration-State Timer Interval Controls the maximum amount of time (T_max) that FCCU can be in Configuration state according to this equation: T_max = $T_{16MHz} \times 2^{(TO + 10)}$ If you put FCCU in Configuration state and this timer interval expires (called a Configuration-state timeout), then FCCU: • Automatically leaves Configuration state and enters Normal state. • Changes the value of the Configuration (FCCU_CFG) register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. For information on the Configuration (FCCU_CFG) register's Configuration-state-timeout value, see FCCU_CFG register bit value sources (N and C) by event. For a list of configuration registers, see Configuration registers. 000: 64 µs ... 111: 8192 µs

74.10.12 IO Control (FCCU_EINOUT)

The FCCU_EINOUT register allows the following operations typically in NORMAL state:

- to control the EOUT[1] output level when the FCCU is configured in "Test1" or "Test0" fault output mode (FCCU_CFG.FOM)
- to control the EOUT[0] output level when the FCCU is configured in "Test1" or "Test2" fault output mode (FCCU_CFG.FOM)
- to observe the EOUT[1:0] signals in input mode

Table 74-9 shows Bi-stable encoding.

Table 74-9. Bi-stable encoding

Mode = FCCU_CFG.FOM	EOUT[0]	EOUT[1]
Test1	output	output
Test2	output	input
Test0	input	output

NOTE

Due to the resynchronization stage of the EOUT interface, there is a latency of a few CLKSAFE cycles following a write/read operation of the FCCU_EINOUT register.

Table 74-10. EOUTX bit encoding

Register		Signals		Description
EINOUT.EOUT0	EINOUT.EOUT0	FCCU_EOUT1	FCCU_EOUT0	
0	0	1 xor PS	0 xor PS	Opposite phase
0	1	0 xor PS	1 xor PS	Opposite phase
1	0	0 xor PS	0 xor PS	In phase
1	1	1 xor PS	1 xor PS	In phase

Address: 0h base + BCh offset = BCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0										EIN1	EIN0	0		EOUTX	
W																
Reset	0	0	0	0	0	0	0	0	0	0	x*	x*	0	0	0	0

* Notes:
- x = Undefined at reset.

FCCU_EINOUT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 EIN1	Error Input 1 Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the state of the EIN1 signal. 0 Low 1 High
27 EIN0	Error Input 0 Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the state of the EIN0 signal. 0 Low 1 High
28–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 EOUTX	EOUT protocol is SW controlled in NORMAL/ALARM state and HW controlled in FAULT state (dual rail, in phase toggling with programmable delay) a HW interrupt trigger (IRQ_EOUT) asserted by the FCCU (EOUT counter saturation) when in NORMAL/ALARM state in order to generate the EOUT protocol via SW. As alternative an external eTMR can be optionally used. EOUT dual rail protocol: NORMAL/ALARM : e.g. 10 ms toggling in opposite phase FAULT : e.g. 10 ms toggling in phase starting after a programmable delay T_tgl, slow switching mode (SM=0) FAULT : fast switching mode (SM =1) in phase (no periodic toggling). For details, please refer to Option 3 See Table 74-10. **NOTE:** When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value.

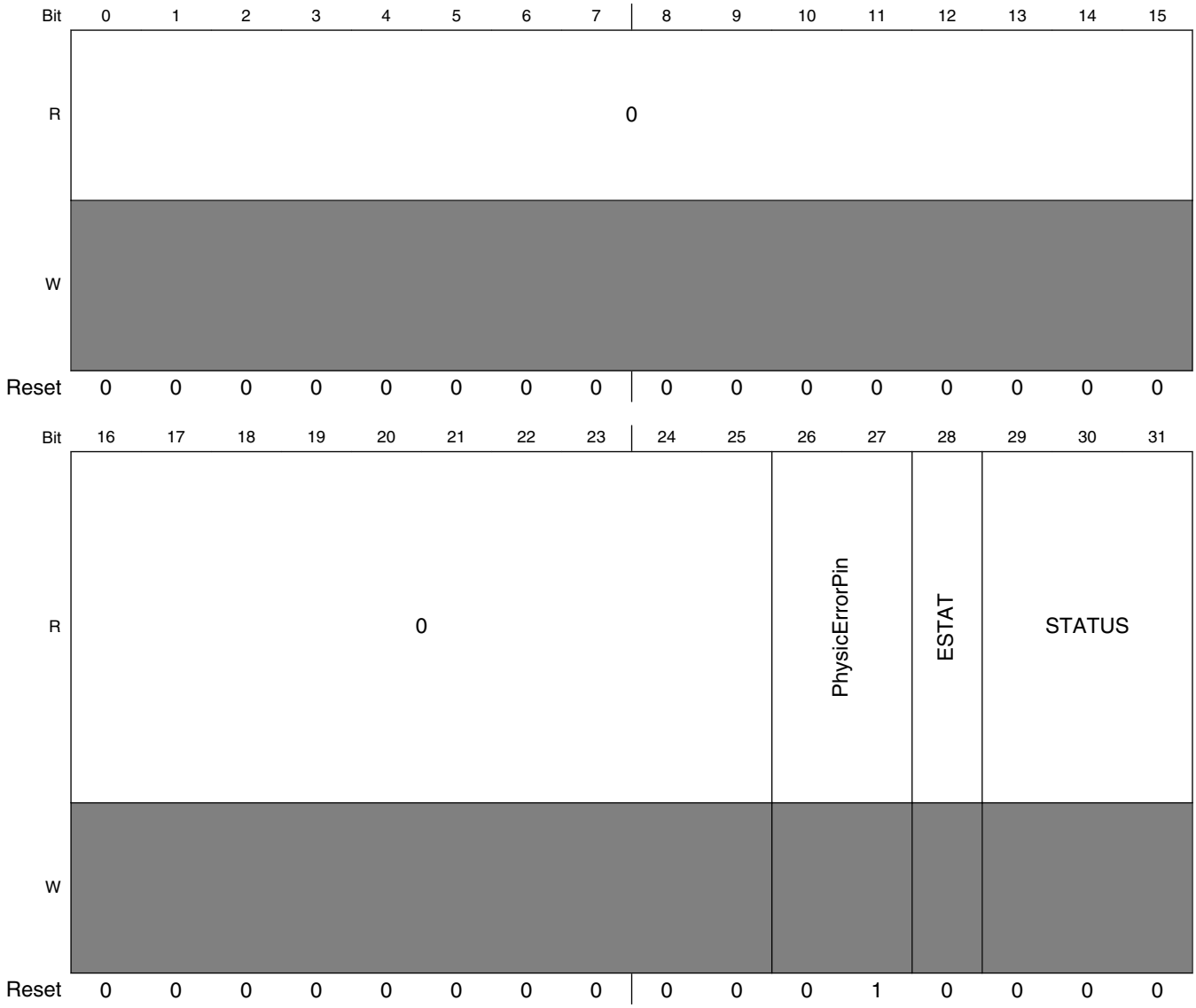
74.10.13 Status (FCCU_STAT)

The FCCU_STAT register includes the FCCU status for debugging/test purposes. The FCCU finite state machine operates by the RC oscillator clock asynchronous with the system clock. The FCCU status read operation requires a safe mechanism operated by a HW/SW synchronization sequence. The SW application executes a FCCU status read operation by the following sequence:

1. Set the OP3 operation into the FCCU_CTRL.OPR field.

2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU status (FCCU_STAT register).

Address: 0h base + C0h offset = C0h



FCCU_STAT field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26–27 PhysicErrorPin	Fault Output (EOUT) States Applies only after successfully running an OP3 operation (OPR) and only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the states that FCCU is driving on the EOUT signals. 00 EOUT1 is low; EOUT0 is low. 01 EOUT1 is low; EOUT0 is high.

Table continues on the next page...

FCCU_STAT field descriptions (continued)

Field	Description
	10 EOUT1 is high; EOUT0 is low. 11 EOUT1 is high; EOUT0 is high.
28 ESTAT	Fault State Applies only after successfully running an OP3 operation (OPR). Indicates whether FCCU is in Fault state. 0 Not in Fault state (in Normal, Alarm, or Configuration state) 1 In Fault state
29–31 STATUS	FCCU State Applies only after successfully running an OP3 operation (OPR). Indicates FCCU's state. 000 Normal 001 Configuration 010 Alarm 011 Fault 100 Reserved 101 Reserved 110 Reserved 111 Reserved

74.10.14 NA Freeze Status (FCCU_N2AF_STATUS)

The N2AF_STATUS register contains a unique code to identify the "noncritical fault" source (fccu_ncf[x]) that caused the state transition from the NORMAL state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

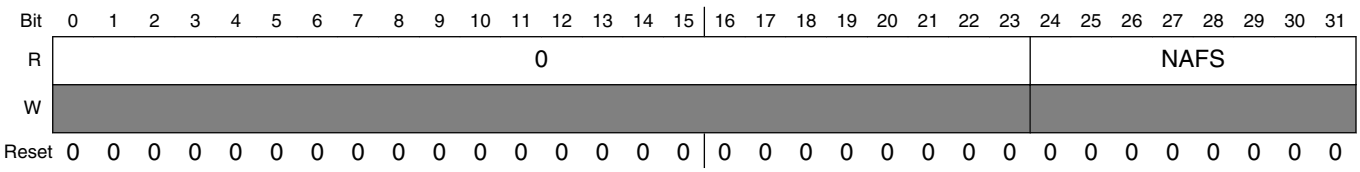
The SW application executes the N2AF_STATUS read operation by the following sequence:

1. Set the OP4 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2AF_STATUS register.

The SW application executes the N2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + C4h offset = C4h



FCCU_N2AF_STATUS field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–31 NAFS	Normal to Alarm Frozen Status 00000000 No transition from NORMAL to ALARM state 00000001 NORMAL to ALARM state transition cause => fccu_ncf[0] fault 00000010 NORMAL to ALARM state transition cause => fccu_ncf[1] fault 00000011 NORMAL to ALARM state transition cause => fccu_ncf[2] fault ... 10000000 NORMAL to ALARM state transition cause => fccu_ncf[127]/ fault 11111111 NORMAL to ALARM state transition cause => multiple fccu_ncf[]/ faults

74.10.15 AF Freeze Status (FCCU_A2FF_STATUS)

The A2FF_STATUS register contains a unique code to identify the timeout trigger (noncritical fault) that caused the state transition from the ALARM state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the A2FF_STATUS read operation by the following sequence:

1. Set the OP5 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the A2FF_STATUS register.

The SW application executes the FCCU_AFFS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).

All the freeze registers are cleared by this operation.

Address: 0h base + C8h offset = C8h

Bit	0–15	16–21	22–23	24–31
R	0	0	AF_SRC	AFFS
W				
Reset	0	0	0	0

FCCU_A2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 AF_SRC	Fault source 00 No fault 01 Reserved 10 "Noncritical" fault 11 Multiple and/or "noncritical" faults
24–31 AFFS	Alarm to Fault Frozen Status 00h: No transition from ALARM to FAULT state 01h: ALARM to FAULT state transition cause => fccu_ncf[0]/ fault timeout fault 02h: ALARM to FAULT state transition cause => fccu_ncf[1]/ fault timeout fault 03h: ALARM to FAULT state transition cause => fccu_ncf[2]/ fault timeout fault ... 80h: ALARM to FAULT state transition cause => fccu_ncf[127]/ fault timeout fault FFh: ALARM to FAULT state transition cause => multiple fccu_ncf[]/ faults timeout faults

74.10.16 NF Freeze Status (FCCU_N2FF_STATUS)

The N2FF_STATUS register contains a unique code to identify the source of the noncritical fault that caused the state transition from the NORMAL state to the FAULT state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes. The SW application executes the N2FF_STATUS read operation by the following sequence:

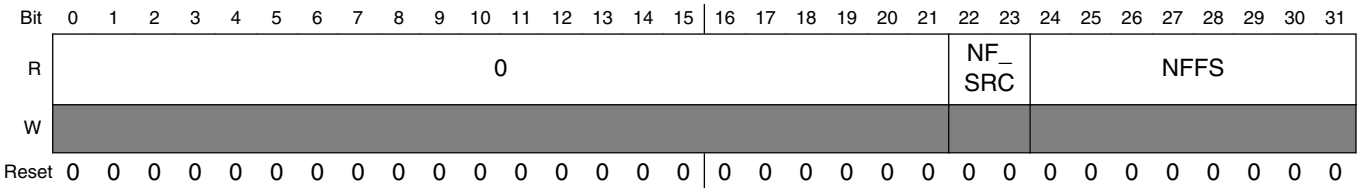
1. Set the OP6 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the N2FF_STATUS register.

The SW application executes the N2FF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.

2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + CCh offset = CCh



FCCU_N2FF_STATUS field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22–23 NF_SRC	NF_SRC: Fault source 00 No fault 01 Reserved 10 "Noncritical" fault 11 Multiple "noncritical" faults
24–31 NFFS	Normal to Fault Frozen Status 00h: No transition from NORMAL to FAULT state 01h: NORMAL to FAULT state transition cause => fccu_ncf[0] fault 02h: NORMAL to FAULT state transition cause => fccu_ncf[1] fault 03h: NORMAL to FAULT state transition cause => fccu_ncf[2] fault ... 80h: NORMAL to FAULT state transition cause => fccu_ncf[127] fault FFh: NORMAL to FAULT state transition cause => multiple fccu_ncf[] faults

74.10.17 FA Freeze Status (FCCU_F2A_STATUS)

The F2AF_STATUS register contains a unique code to identify the source of the noncritical fault (fccu_ncf[x]) that caused the state transition from the FAULT state to the ALARM state. In case of multiple fault conditions the fault source cannot be identified. This register is for test/debug purposes.

The SW application executes the F2AF_STATUS read operation by the following sequence:

1. Set the OP7 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the F2AF_STATUS register.

The SW application executes the F2AF_STATUS clear operation by the following sequence:

1. Set the OP13 operation into the FCCU_CTRL.OPR field.
2. Wait for the completion of the operation (FCCU_CTRL.OPS field). (All the freeze registers are cleared by this operation.)

Address: 0h base + D0h offset = D0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																							FAFS								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_F2A_STATUS field descriptions

Field	Description
0–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–31 FAFS	Fault to Normal Frozen Status 00h: No transition from FAULT to ALARM state 01h: FAULT to ALARM state transition cause => fccu_ncf[0]/ fault 02h: FAULT to ALARM state transition cause => fccu_ncf[1]/ fault 03h: FAULT to ALARM state transition cause => fccu_ncf[2]/ fault ... 80h: FAULT to ALARM state transition cause => fccu_ncf[127]/ fault FFh: FAULT to ALARM state transition cause => multiple fccu_ncf[] faults

74.10.18 Noncritical Fault Fake (FCCU_NCFF)

The FCCU_NCFF register contains a unique code to set a noncritical fault in mutually exclusive mode by the external FAULT interface (signal setting). It allows the SW emulation of the noncritical faults, by the injection of the fault directly in the FAULT root, in order to verify the entire path and reaction. The reaction following a fake noncritical fault cannot be masked. The FCCU_NCFF is a write-only register with a set of codes corresponding to each noncritical fault injection.

Address: 0h base + DCh offset = DCh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																															
W																										FNCFC						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NCFF field descriptions

Field	Description
0–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–31 FNCFC	FNCFC/ : Fake noncritical fault code **NOTE:** Only writing to this register, fake fault injection occurs, writing 00 and default value being zero give different results. 00h: Fake noncriticalfault injection at noncriticalfault source 0 01h: Fake noncriticalfault injection at noncriticalfault source 1 02h: Fake noncriticalfault injection at noncriticalfault source 2 ... 7Fh: Fake noncriticalfault injection at noncriticalfault source 127

74.10.19 IRQ Status (FCCU_IRQ_STAT)

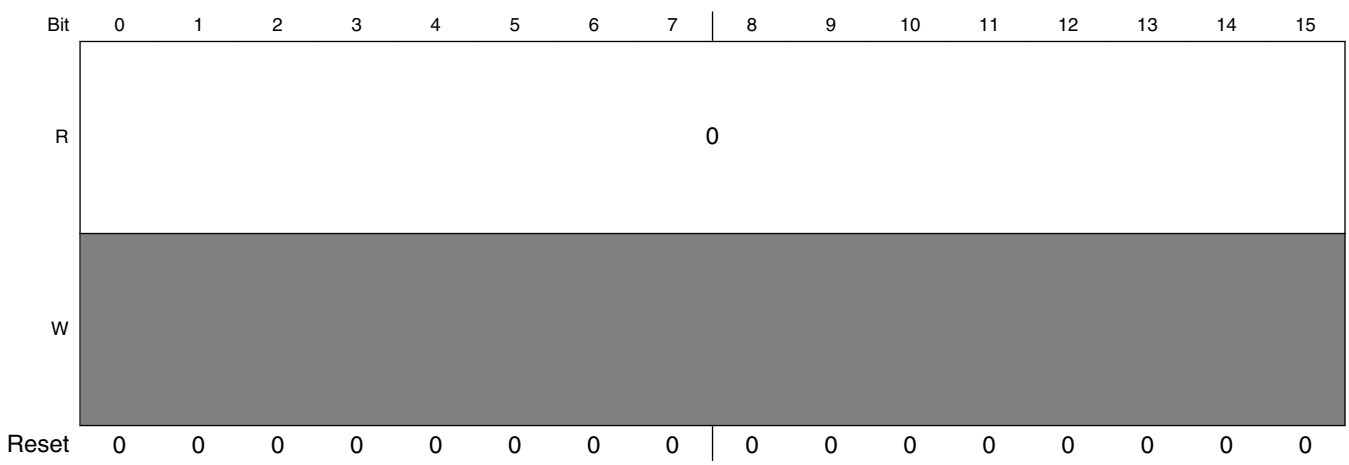
The FCCU_IRQ_STAT register provides the FCCU interrupt status related to the following events:

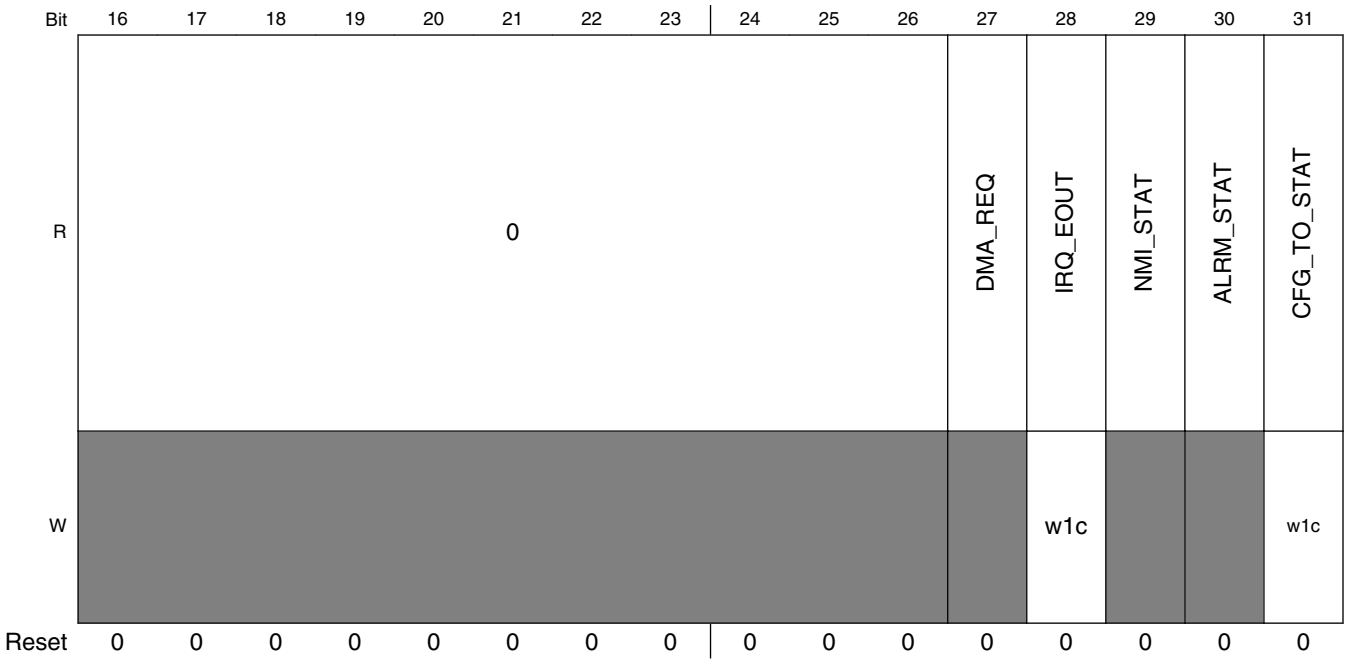
- Configuration timeout error
- Alarm interrupt
- NMI interrupt

The configuration timeout interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set. It is cleared when a 1 is written to the CFG_TO_STAT bit.

The NMI and ALARM interrupts are asserted and cleared according to the FCCU state. The status bits of the FCCU_IRQ_STAT trace the status of the related interrupt lines.

Address: 0h base + E0h offset = E0h





FCCU_IRQ_STAT field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 DMA_REQ	Status bit asserted when EOUT counter = T_TRG in NORMAL/ALARM mode (DMA_REQ assertion). For details, please refer to Option 1, Option 2 section. Cleared on DMA_ACK assertion. **NOTE:** For more information: In the CFG register, see the FOME field description for the setting 10b. **NOTE:** SPI frame is DMA controlled following a DMA request (DMARQ_EOUT) asserted by the FCCU (EOUT counter saturation) when in NORMAL/ALARM state. An interrupt by SPI/DMA engine is asserted in order to update the data packet for the next watchdog refresh SPI controlled only via DMA in time division mode..
28 IRQ_EOUT	Interrupt status bit asserted when EOUT counter = T_TRG in NORMAL/ALARM mode. Usable to assert an interrupt (if FCCU_IRQ_EN[IRQ_EOUT_EN] = 1) and to control via SW the EOUTx pins in NORMAL/ALARM mode. **NOTE:** For more information: In the CFG register, see the FOME field description for the settings 01b and 11b. **NOTE:** SPI frame is SW controlled following a HW interrupt trigger (IRQ_EOUT) asserted by the FCCU (EOUT counter saturation) when in NORMAL/ALARM state.. For details, please refer to Option1 , Option 2 section. **NOTE:** EOUT protocol is SW controlled in NORMAL/ALARM state and HW controlled in FAULT state (dual rail, in phase toggling with programmable delay) a HW interrupt trigger (IRQ_EOUT) asserted by the FCCU (EOUT counter saturation) when in NORMAL/ALARM state in order to generate the EOUT protocol via SW. As alternative an external eTMR can be optionally used. EOUT dual rail protocol: NORMAL/ALARM : e.g. 10 ms toggling in opposite phase FAULT : e.g. 10 ms toggling in phase starting after a programmable delay T_tgl, slow switching mode (SM=0) FAULT : fast switching mode (SM =1) in phase (no periodic toggling). For details, please refer to Option 3 section.

Table continues on the next page...

FCCU_IRQ_STAT field descriptions (continued)

Field	Description
29 NMI_STAT	NMI Interrupt Status 0 NMI interrupt is OFF 1 NMI interrupt is ON
30 ALRM_STAT	Alarm Interrupt Status 0 Alarm interrupt is OFF 1 Alarm interrupt is ON
31 CFG_TO_STAT	Configuration Timeout Status 0 No configuration timeout error 1 Configuration timeout error

74.10.20 IRQ Enable (FCCU_IRQ_EN)

The FCCU_IRQ_EN register defines the FCCU interrupt enable register related to the following event:

- Configuration timeout error

The configuration timeout interrupt is asserted if the CFG_TO_STAT bit of the FCCU_IRQ_STAT register is set and the CFG_TO_IEN bit of the FCCU_IRQ_EN register is also set.

Address: 0h base + E4h offset = E4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													0		
W													IRQ_EOUT_EN			CFG_TO_IEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_IRQ_EN field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 IRQ_EOUT_EN	Enables the FCCU_IRQ_STAT[IRQ_EOUT] 0 IRQ_EOUT disabled 1 IRQ_EOUT enabled
29–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 CFG_TO_IEN	Configuration Timeout Interrupt Enable 0 Configuration timeout interrupt disabled 1 Configuration timeout interrupt enabled

74.10.21 XTMR (FCCU_XTMR)

The FCCU_XTMR register contains the read values of the Alarm or Watchdog Timer. These timers are clocked by CLKSAFE.

ETMR is the EOUT pin in logic 0 counter. After EOUT has been set to low (by SW or by a real fault), it goes high at the end of the T_min period from the last fault trigger.

The SW application executes the timer read operation by the following sequence:

1. Set any of the following operations into the FCCU_CTRL.OPR field:
   - OP17
   - OP19
   - OP20
2. Wait for the completion of the operation (FCCU_CTRL.OPS field).
3. Read the FCCU_XTMR register.

Table 74-11 shows Timer state/value.

Table 74-11. Timer state/value

TIMER	CONFIG state	NORMAL state	ALARM state	FAULT state
ALARM	00000000h	Initial value	Running	Idle/End of count
CFG	Running	0001FFFFh	0001FFFFh	0001FFFFh
ETMR	00000000h	00000000h	00000000h	Running

Address: 0h base + E8h offset = E8h

Bit	0–31
R	XTMR
W	
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

FCCU_XTMR field descriptions

Field	Description
0–31 XTMR	Alarm/Watchdog The current timer value is measured in CLKSAFE cycles. These bits can be read by the software.

74.10.22 Mode Controller Status (FCCU_MCS)

Indicates the last four chip modes—as defined by the MC_ME module—and whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured each mode.

Address: 0h base + ECh offset = ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VL3	FS3	0		MCS3				VL2	FS2	0		MCS2			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VL1	FS1	0		MCS1				VL0	FS0	0		MCS0			
W																
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value is chip-specific. See the chip-specific FCCU information. (The MC_ME module changes the value of the register immediately after the chip leaves RESET mode.)

FCCU_MCS field descriptions

Field	Description
0 VL3	Valid 3—Indicates whether MCS3 and FS3 are valid. 0 Invalid 1 Valid

Table continues on the next page...

FCCU_MCS field descriptions (continued)

Field	Description
1 FS3	Fault Status 3 Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS3. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 Not in Fault state or NMIOUT not asserted 1 In Fault state and NMIOUT asserted
2–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MCS3	Mode Controller State 3—Indicates the fourth most recent chip mode.
8 VL2	Valid 2—Indicates whether MCS2 and FS2 are valid. 0 Invalid 1 Valid
9 FS2	Fault Status 2 Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS2. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 Not in Fault state or NMIOUT not asserted 1 In Fault state and NMIOUT asserted
10–11 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–15 MCS2	Mode Controller State 2—Indicates the third most recent chip mode.
16 VL1	Valid 1—Indicates whether MCS1 and FS1 are valid. 0 Invalid 1 Valid
17 FS1	Fault Status 1 Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS1. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 Not in Fault state or NMIOUT not asserted 1 In Fault state and NMIOUT asserted
18–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FCCU_MCS field descriptions (continued)

Field	Description
20–23 MCS1	Mode Controller State 1—Indicates the second most recent chip mode.
24 VL0	Valid 0—Indicates whether MCS0 and FS0 are valid. 0 Invalid 1 Valid
25 FS0	Fault Status 0 Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS0. **NOTE:** This field might occasionally be inaccurate because the MC_ME module provides the chip modes to FCCU in synchronization with CLKSYS, but FCCU captures the chip modes in synchronization with CLKSAFE. 0 Not in Fault state or NMIOUT not asserted 1 In Fault state and NMIOUT asserted
26–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 MCS0	Mode Controller State 0—Indicates the most recent (current) chip mode.

74.10.23 Transient Configuration Lock (FCCU_TRANS_LOCK)

Writable only by code running in Supervisor mode. Temporarily locks and unlocks the configuration. Locking the configuration prevents FCCU from entering Configuration state.

Address: 0h base + F0h offset = F0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W																								TRANSKEY								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

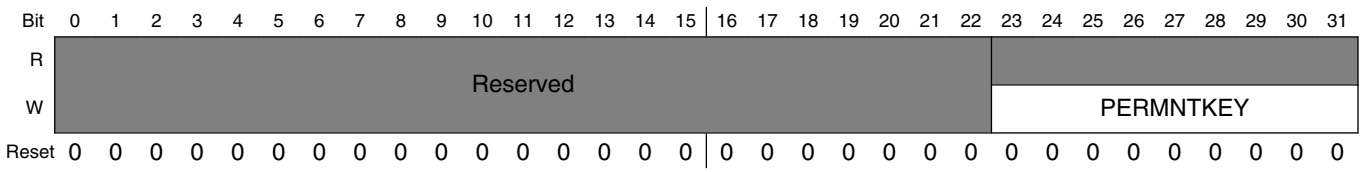
FCCU_TRANS_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved.
23–31 TRANSKEY	Transient Configuration Lock Writable only by code running in Supervisor mode. Temporarily locks and unlocks the configuration. Locking the configuration prevents FCCU from entering Configuration state. BCh: Unlock. Any other value: Lock.

74.10.24 Permanent Configuration Lock (FCCU_PERMNT_LOCK)

Writable only by code running in Supervisor mode. Permanently locks the configuration, which prevents FCCU from entering Configuration state until FCCU is reset.

Address: 0h base + F4h offset = F4h



FCCU_PERMNT_LOCK field descriptions

Field	Description
0–22 Reserved	This field is reserved.
23–31 PERMNTKEY	Permanent Configuration Lock Writable only by code running in Supervisor mode. Permanently locks the configuration, which prevents FCCU from entering Configuration state until FCCU is reset. FFh: Lock. Any other value: Do nothing.

74.10.25 Delta T (FCCU_DELTA_T)

The FCCU_DELTA_T register is used for programming the value of delta_T constant, in microseconds.

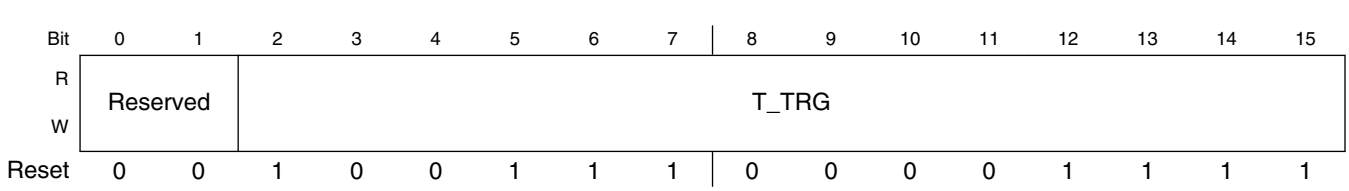
NOTE

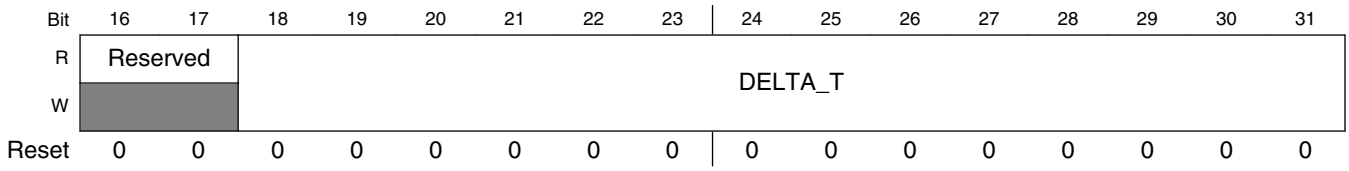
This register can be written only when the FCCU is in CONFIG state.

NOTE

Reserved bits should always be written as all 0's.

Address: 0h base + F8h offset = F8h





FCCU_DELTA_T field descriptions

Field	Description
0–1 Reserved	This field is reserved.
2–15 T_TRG	IRQDMA trigger time In FAULT mode, if CFG[SM] is 0, the EOUT semi-period is T_TRG + 1 [µs]. In NORMAL/ALARM mode, if T_TRG > 0, T_TRG is the IRQ_EOUTDMA_REQ interrupt time base. Min = 2 µs, Max = 16.385 µs, Resolution = 1 µs **NOTE:** EOUT protocol is SW controlled in NORMAL/ALARM state and HW controlled in FAULT state (dual rail, in phase toggling with programmable delay) a HW interrupt trigger (IRQ_EOUT) asserted by the FCCU (EOUT counter saturation) when in NORMAL/ALARM state in order to generate the EOUT protocol via SW. As alternative an external eTMR can be optionally used. EOUT dual rail protocol: NORMAL/ALARM : e.g. 10 ms toggling in opposite phase FAULT : e.g. 10 ms toggling in phase starting after a programmable delay T_tgl, slow switching mode (SM=0) FAULT : fast switching mode (SM =1) in phase (no periodic toggling). For details, please refer to Option 3 section.
16–17 Reserved	This field is reserved.
18–31 DELTA_T	Bistable Minimum Fault-Output (EOUT) Faulty Interval Applies only to Bistable fault-output mode (FOM). Controls the minimum amount of time (T_min) that the fault-output (EOUT) timer runs according to this equation: T_min = 250 µs + DELTA_T The following values and meanings are for DELTA_T: 00_0000_0000_0000: 0 µs 00_0000_0000_0001: 1 µs … 11_1111_1111_1110: 16,382 µs 11_1111_1111_1111: 16,383 µs **NOTE:** The durations shown for the DELTA_T values are for CLKSAFE clock signals running at exactly 16 MHz. However, the CLKSAFE signals are sourced from an internal chip clock signal (for example, IRC)—or an integer division of it (see the chip-specific FCCU information)—that has a trimmed frequency variation. Therefore, the DELTA_T durations may vary. See your chip's data sheet for the trimmed frequency variation (for example, δF_{var}).

74.10.26 IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*)

These registers enable the corresponding IRQ alarm.

IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*) register (value of n)	Offset	IRQENx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	0FCh	31	0
1	100h	63	32
2	104h	95	64

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + FCh offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IRQENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_IRQ_ALARM_EN*n* field descriptions

Field	Description
0–31 IRQENx	IRQ alarm enable For the mapping of the IRQENx fields among registers, see the earlier table in this register description. 0 Alarm is disabled for error source x. 1 Alarm is enabled for error source x.

74.10.27 NMI Enable (FCCU_NMI_EN*n*)

These registers enable the NMI.

Table 74-12 shows FCCU NMI enable register channels.

Table 74-12. FCCU NMI enable register channels

Address offset	Register name	Channel range (x) (bit location [0:31])
10Ch	FCCU_NMI_EN0	NMIENE[31:0]
110h	FCCU_NMI_EN1	NMIENE[63:32]
114h	FCCU_NMI_EN2	NMIENE[95:64]

NOTE

These registers can be written only when the FCCU is in CONFIG state.

Address: 0h base + 10Ch offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NMIENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_NMI_EN*n* field descriptions

Field	Description
0–31 NMIENx	NMI enable See Table 74-12 for register offset to channel number relationship. 0 NMI is disabled for error (NCF) source x. 1 NMI is enabled for error (NCF) source x.

74.10.28 Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*)

Writable only when FCCU is in Configuration state. Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). When FCCU is configured for Bistable fault-output mode (FOM), controls whether fault-output (EOUT) signaling is enabled for the associated noncritical fault channel (x). (For other fault-output modes, fault-output signaling is always enabled, regardless of the value of this field.)

When FCCU is configured for a fault-output mode other than Bistable, controls whether FCCU asserts the FIF signal when a fault occurs on the associated noncritical fault channel (x). (When FCCU is configured for Bistable fault-output mode, FCCU always asserts the FIF signal when a fault occurs on the channel, regardless of the value of this field.)

Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*) register (value of n)	Offset	EOUTENx fields (value of x)	
		Most significant (leftmost) bit	Least significant (rightmost) bit
0	11Ch	31	0
1	120h	63	32
2	124h	95	64

Address: 0h base + 11Ch offset + (4d × i), where i=0d to 2d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EOUTENx																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FCCU_EOUT_SIG_EN*n* field descriptions

Field	Description
0–31 EOUTENx	Bistable Fault-Output (EOUT) Mode Signaling Enable x Writable only when FCCU is in Configuration state. Applies only when the EOUT signals are active (FCCU_SET_AFTER_RESET). When FCCU is configured for Bistable fault-output mode (FOM), controls whether fault-output (EOUT) signaling is enabled for the associated noncritical fault channel (x). (For other fault-output modes, fault-output signaling is always enabled, regardless of the value of this field.) When FCCU is configured for a fault-output mode other than Bistable, controls whether FCCU asserts the FIF signal when a fault occurs on the associated noncritical fault channel (x). (When FCCU is configured for Bistable fault-output mode, FCCU always asserts the FIF signal when a fault occurs on the channel, regardless of the value of this field.) For the mapping of the EOUTENx fields among registers, see the earlier table in this register description. 0 Disabled 1 Enabled

74.10.29 Configuration registers

74.10.29.1 Definition

Configuration registers are registers that:

- Let you configure FCCU's Alarm-state timer interval, fault channels, and fault-output (EOUT) signals.
- You can write to only when the configuration is not locked (see Lock and unlock the configuration) and FCCU is in Configuration state (see Put FCCU in Configuration or Normal state).
- Save the values you write to them while FCCU is in Configuration state only after you manually put FCCU in Normal state. If FCCU automatically leaves Configuration state and enters Normal state because the configuration-timer interval (TO) expires (called a Configuration-state timeout), FCCU changes the value of the FCCU_CFG register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. For information on the Configuration-state timeout value, see FCCU_CFG register bit value sources (N and C) by event.

74.10.29.2 The configuration registers

The following registers are the configuration registers, listed in offset order from lowest to highest:

- Configuration (FCCU_CFG)
- Noncritical Fault Configuration (FCCU_NCF_CFG*n*)

- Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*)
- Noncritical Fault Enable (FCCU_NCF_E*n*)
- Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*)
- Noncritical Fault Timeout (FCCU_NCF_TO)
- Delta T (FCCU_DELTA_T)
- IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*)
- NMI Enable (FCCU_NMI_EN*n*)
- Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*)

74.10.30 FCCU_CFG register bit value sources (N and C) by event

Certain events affect the bit values of the Configuration (FCCU_CFG) register; when one of these events occurs, the bit value is either unaffected, or FCCU changes it to one of the following:

- N: a chip-specific (or, for some chips, user-specified) value provided by the bit's associated NVMCFG signal
- C: a chip-specific value provided by the bit's associated constant inside the module

For each bit in the Configuration (FCCU_CFG) register, this table shows the associated NVMCFG signal, if any, and indicates the source of the bit value for the events that affect it. For specific values of *N* and *C* for each bit, see the chip-specific FCCU information.

Bit	NVMCFG signal	FCCU reset	Configuration-state timeout	OP31 operation
0	21	N	C	N
1	20	N	C	N
2	19	N	C	N
3	18	N	C	N
4	17	N	C	N
5	16	N	C	N
6	15	N	C	N
7	—	C	C	(Unaffected)
8	—	C	C	(Unaffected)
9	—	C	C	(Unaffected)
10	—	C	C	(Unaffected)
11	—	C	C	(Unaffected)
12	—	C	C	(Unaffected)
13	—	C	C	(Unaffected)

Table continues on the next page...

Bit	NVMCFG signal	FCCU reset	Configuration-state timeout	OP31 operation
14	—	C	C	(Unaffected)
15	—	C	C	(Unaffected)
16	14	N	C	N
17	13	N	C	N
18	12	N	C	N
19	—	C	C	(Unaffected)
20	11	N	C	N
21	10	N	C	N
22	9	N	C	N
23	8	N	C	N
24	7	N	C	N
25	6	N	C	N
26	5	N	C	N
27	4	N	C	N
28	3	N	C	N
29	2	N	C	N
30	1	N	C	N
31	0	N	C	N

74.11 Use cases and limitations

74.11.1 Configuration guidelines

Follow these guidelines to configure FCCU:

- If you want FCCU to react to a fault on a noncritical fault channel:
  - Enable the channel (NCFEx).
  - Enable at least one type of Fault-state reaction for the channel: chip reset (NCFSCx), fault-output (EOUT) signaling (EOUTENx), or nonmaskable interrupt (NMIENx).
  - If you enable chip reset as the type of Fault-state reaction for the channel (NCFSCx), enable either Alarm state (NCFTOEx) or at least one other type of Fault-state reaction for the channel: fault-output (EOUT) signaling (EOUTENx) or nonmaskable interrupt (NMIENx).
  - If you enable Alarm state for the channel (NCFTOEx), enable the Alarm-state reaction (IRQENx).

NOTE

If you enable a noncritical fault channel but disable all reactions for that channel, FCCU changes state if necessary but doesn't perform any reaction because reactions are disabled. If you enable reactions for a noncritical fault channel but disable that channel, and FCCU is in Normal state when a fault occurs on the channel, FCCU doesn't enter Alarm or Fault state and therefore doesn't perform any reaction.

74.11.2 Recommendations to configure FCCU

1. After a power on, 'destructive', or long external reset, where both system and FCCU are reset, the following steps could be followed to configure FCCU:
   a. Check and clear any pending fault status
   b. Verify FCCU is in NORMAL state, else repeat step(a) above
   c. Configure FCCU
2. After a short external or any 'functional' reset of the system, arising out of a reset request from FCCU or other sources, the following steps could be followed to reconfigure FCCU:
   a. If active, wait for the Error out T_min to expire
   b. Check and clear fault status
   c. Error pin moves to "non faulty" state, once fault status is cleared and T_min expires
   d. Verify FCCU is in NORMAL state, else repeat step(a) above
   e. Read and verify value in FCCU_NCF_Ex
   f. Reconfigure FCCU, if necessary

Chapter 75
Self-Test Control Unit (STCU2)

75.1 Chip-specific STCU2 information

75.1.1 STCU configuration

The Self-Test Control Unit (STCU2) controls the execution of the built-in self-test of the MPC5746C.

The built-in self-test supports in the application:

- Memory (RAM/ROM) Built-In Self-Test (MBIST) at startup and shutdown.

The following table summarizes the three self-test modes.

Table 75-1. STCU2 self-test modes

Self-test mode	Description	Availability
Normal self-test mode	MBIST for all memories with a checkerboard/inverted checkerboard algorithm	In startup mode and shutdown mode
Long self-test mode	MBIST for all memories with the reduced RAM and ROM algorithm	Only during shutdown mode
Diagnostic mode	MBIST for all memories with the full RAM and ROM algorithm	Only during shutdown mode

In application mode, the STCU2 initiates self-test execution:

- after Power-on Reset (POR)
- after external reset (if triggered as long-functional reset)
- after destructive reset
- when initiated by software

The STCU2 communicates any fault information to the Fault Control and Collection Unit (FCCU) and to the Reset Generation Module (MC_RGM).

The startup self-test is implemented by NXP. The user can choose either to run normal self-test execution during startup of the device or to bypass the startup self-test execution by programming the appropriate records in the UTEST flash memory.

The shutdown self-test sequence is user programmable and must be started by the user application on demand. This self-test can be executed at less critical times in the application. After the self-test is executed during shutdown, a long functional reset is initiated, and the self-test results are accessible.

The STCU2 includes the SSCM DCF bus to load the self-test parameters from flash memory—consisting of both the test flash memory, which is programmed by NXP, and the UTEST flash, which is user-programmable—during the startup self-test phase.

NOTE

Ensure Flash is in IDLE state (i.e. no Program/Erase) prior to starting Self Test operation.

75.1.2 Supported BIST sequence

NXP has validated and therefore supports only particular STCU2 built-in self-test (BIST) sequences for this chip. For descriptions of the supported BIST sequences for this chip, see the engineering bulletin titled MPC5746C STCU BIST Configuration (document EB820).

75.1.3 Device specific STCU Registers

The STCU2 module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/half-word/word (free byte enables or other operations) on each register
- Any write operation on Security Key register applying a wrong sequence of keys (the two write operations cannot be interleaved with other access to STCU2 registers)
- Any write operation performed on a register when the Security keys have not been applied
- Any write operation performed on read-only registers

Write operations to fields marked as reserved do not generate the transfer error.

75.1.4 STCU register reset values

All reset values for the STCU registers that are device specific are shown in the following table. All other register reset values are shown in the STCU Register description.

NOTE

These are the reset values when startup self-test is not configured by DCF's otherwise these will change as per the DCF configurations of startup self-test.

Table 75-2. STCU register reset values

Register	Register Value
STCU2 Configuration Register (STCU2_CFG)	0x7F00_0000
STCU2 Watchdog Register Granularity (STCU2_WDG)	0x0000_FFFF
STCU2 Error FM Register (STCU2_ERR_FM)	0x0000_0000
STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)	0x0000_0000
STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)	0x0000_0000
STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)	0x0000_0000
STCU2 MBIST Control Register (STCU2_MB_CTRLn)	0x0000_0000

75.1.5 Partition overview

MBIST

The MBIST for the memories on the device is set to separate operations as then the triple SRAM can be started individually. This leads to the flexibility to operate an MBIST on one RAM where as the device is in operation on the other RAM of the device.

An individual MBIST (on a per memory basis) start is beneficial to maintain lower power consumption while MBIST is running in a lower power budget. Each system RAM support a separate MBIST. This allows a separate MBIST execution on the three System RAMs The STCU2 controller handles all memories in the device. Results of the MBIST is reported on the STCU2 to the FCCU.

MBIST Partitions

The following table describes the mapping between the MBIST number, as the STCU2 captures it, and the appropriate memory in the design.

Table 75-3. MBIST mapping

Partition number	Memory in design
0	PRAM0
1	
2	
3	
4	

Table continues on the next page...

Table 75-3. MBIST mapping (continued)

Partition number	Memory in design
5	PRAM1
6	
7	Z4 ICACHE
8	
9	
10	
11	Z4 ITAG
12	Z4 DCACHE
13	
14	
15	
16	Z4 DTAG
17	HSM RAM
18	
19	
20	
21	HSM DCACHE
22	
23	HSM TAG
24	
25	FLEXCAN
26	
27	
28	
29	
30	
31	
32	
33	FLEXRAY DATA
34	DMA
35	FLEXRAY LUTO
36	ENET RX FIFO
37	FLEXRAY ROM
38	BAR ROM
39	FLEXRAY LUT1
40	ENET TX FIFO
41	ENET RXPARSER
42	
43	PRAM1

Table continues on the next page...

Table 75-3. MBIST mapping (continued)

Partition number	Memory in design
44	

75.1.6 STCU On-Line and Off-Line Self-Test sequence preparation

This is the typical mode of using the STCU2 module during the application run. For more details, see Online self test sequence

Before you initiate the execution of an STCU on-line self-test, you must complete the prerequisite procedure. PLL clock needs to be locked at 40 MHz for running On-Line self test sequence in PLL clock. For running self-test sequence on IRC, no additional configuration is required.

75.1.7 AUTOLOCK_VALUE for register write access and DCF_COMPLETION value for DCF write completion

Register write-access watchdog timer refers to two clock-cycle values related to the STCU2's write-access watchdog. The following table provides these values on this chip.

Clock-cycle value	Number of STCU2 clock cycles
AUTOLOCK_VALUE	1024
DCF_COMPLETION	16

75.1.8 Wait time for writing to the online registers

When writing to the online registers, the user must wait for the expiration of the offline key. The time the user should wait is (IPS_CLK)*4096 clock periods. If the IPS_CLK is 66.5 Mhz then the user must wait (1/66.5)*4096 = 61.6 µs.

75.1.9 On-Line reset generation (only MBIST)

The On-Line execution of the MBIST does not generate any reset from STCU2 at the end of the execution. This is because the software application has to manage all the possible consequences related to the write operations performed by the MBIST execution other

than preventing any read/write operation during the Self-Test execution. If necessary, the software must also restore the memory content (only for RAMs) at the end of the Self-Test execution.

NOTE

During online self-test execution, it is possible that a functional reset event does not cause HW abort. In such scenarios, STCU watchdog timeout will generate reset.

NOTE

A functional reset must be generated for proper operation of the chip after online MBIST self-test.

75.2 Introduction

Self Test Control Unit (STCU2) is a comprehensive programmable hardware module that controls the self test sequence applied both during the Off-Line and/or On-Line conditions. It is able to manage by hardware the device's SRAM/ROM Built-In Self Test (MBIST) blocks. The STCU2 includes the SSCM DCF bus to load the Self-test parameters (MBIST scheduling activity, Unrecoverable/Recoverable faults management, PLL management, and so on) from flash memory during the Off-Line Self-Test phase. This interface is able only to write the configuration parameters and start the Self-Test execution once after the STCU2 global reset has been applied. The register access by software is granted by an IPS interface with the purpose to check the results of the Off-Line Self-Test but also to load/check the execution of the On-Line Self-Test.

The STCU2 includes:

- A programmable Watchdog timer (WDG) to check the Self-Test operations (MBIST) have been completed within the assigned time slot or the STCU2 RUN or BYPASS bits have been programmed before Watchdog time-out
- The PLL direct control during the Off-Line Self-Test sequence
- Two sets of Result registers to collect the MBIST result during Off/On-Line Self-Test separately

To increase the security capability of this module, a different Security Key sequence is also required during Off/On-Line Self-Test in order to unlock the write access to the STCU2 registers. The STCU2 also implements two power-saving mechanisms to switch off:

- The global STCU2 clock after Self-Test sequence is completed
- The register ITF clock when the WDG time-out is detected after a Double Security Key sequence is applied during the application run (available only when IPS write access is enabled otherwise the clock is off).

75.3 Main features

- Double registers interface (SSCM DCF with priority over IPS)
- SSCM DCF write one time register interface
- IPS read and write (depends on SSCM enable parameters and STCU2_CFG.WRP bit) register interface
- Programmable scheduler for MBIST execution
- MBIST concurrent/sequential execution
- On-Line hardware Self-Test sequence abort capability
- Programmable Internal clock prescaler to reduce Internal and MBIST TCK clocks
- Programmable PLL direct control during Off-Line Self-test sequence
- PLL Lock signal monitoring during Off/On-Line Self Test
- Programmable WDG Timer for Self-Check MBIST Time Execution
- Hard Coded WDG time-out to flag off-line initialization faults
- Hard Coded WDG time-out to Auto-lock STCU2 write access after Double-Key sequence
- SRAM/ROM MBIST Off-Line status flags
- STCU2 internal errors Off-Line status flag
- SRAM/ROM MBIST On-Line status flags
- STCU2 internal errors On-Line Status flag
- On-Line hardware abort status flag
- SRAM/ROM MBIST Off/On-Line programmable failure severity - unrecoverable fault/recoverable fault (UF/RF)
- STCU2 internal errors Off-On/Line programmable failure severity - unrecoverable fault/recoverable fault (UF/RF)

- STCU2 errors (UF/RF) lines for FCCU signaling
- Redundant UF/RF generation logic to improve reliability
- FCCU UF/RF faults injection mechanism
- Self-Test Bypass capability
- Global write register's protection mechanism based on Double Security Key
- STCU2 Global Auto Power-saving when Self-Test is completed, bypass mode is selected or WDT time-out is detected
- STCU2 ITF/WDG clock wake-up mechanism when software application writes Double Security Key sequence (available when STCU2_CFG.WRP bit is active)
- STCU2 ITF/WDG Auto Power-saving when Hard Coded WDG time-out is detected after software application writes Double Security Key sequence (available when STCU2_CFG.WRP bit is active)

75.4 Block diagram

The top level diagram of the STCU2 module is given in Figure 75-1.

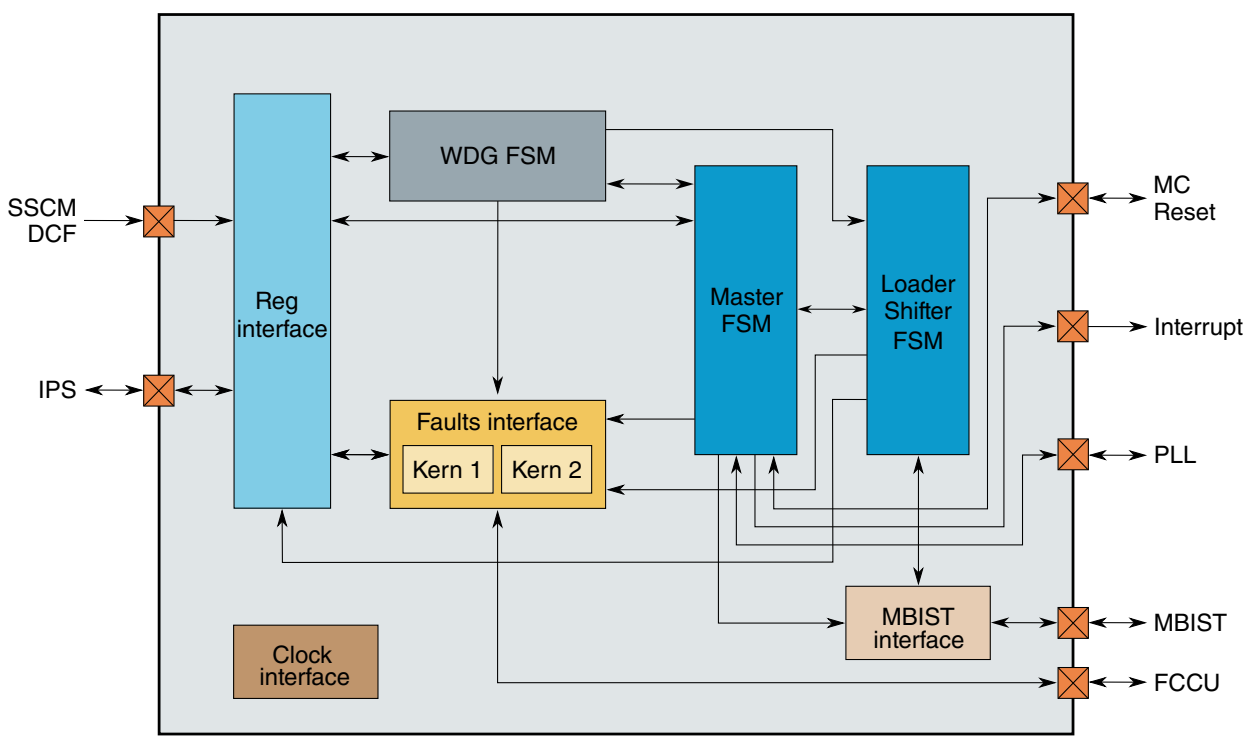


Figure 75-1. STCU2 block diagram

The STCU2 module includes the following sub-modules:

- ***Reg interface***: This module includes the registers, the security key logic, the IPS and SSCM DCF bus interface
- ***Fault interface***: This module collects the fault conditions related to MBIST execution and STCU2 internal failures and, depending on the UF/RF configuration of each one of them, sets the global STCU2 UF or RF flag. It also manages the UF/RF fault lines and the Set/Clear injection mechanism provided by the FCCU. To improve the intrinsic reliability of this critical logic, the generation logic is duplicated.
- ***Clock Interface***: This module manages the internal and the MBIST TCK clock prescaler, the internal Clock Gating Power Saving and the wake-up clock feature.
- ***WDG FSM***: This module includes two different Watchdogs. The first one is Hard Coded and is used to Auto Lock the STCU2 access forcing a reset condition on the Double Security Key registers while the second one has a double functionality. After a reset event initializes the STCU2, it is used as Hard Coded Watchdog time-out while during MBIST run it is used as programmable Watchdog timer to check the MBIST have been completed in the assigned MBIST time slot.
- ***STCU2 Master FSM***: This module includes the main FSM of the STCU2 used to coordinate and schedule all the operations performed during the Self-Test Sequence.
- ***STCU2 Load Shifter FSM***: This module includes the common shifter register and the related state machine used to program the MBIST registers and read back the data to be checked at the end of each Test operation.
- ***MBIST interface***: This module includes the interface between the MBIST controller and STCU2 controllers.

75.5 IPS bus interface

The IPS bus interface is a slave bus used for configuration purposes via CPU. The following bus read operations (contiguous byte enables) are supported:

- Word (32 bits) data read operations to any registers
- Low and high half-words (16 bits, data[31:16] or data[15:0]) data read operations to any registers
- Byte(8 bits, data[31:24] or data[23:16] or data[15:8] or data[7:0]) data read operations to any registers
- Any other operation (free byte enables or other operations) has to be avoided.

The same operations are also supported in write mode only in the following scenario:

- The Self-Test sequence parameters are loaded before Off-Line Self-test is executed
- The IPS access on a specific register is allowed by software only when the bit STCU2_CFG.WRP is cleared
- In all the other conditions the write operations are not supported and all the STCU2 registers can be only read. Only exceptions are the following read/write bits: STCU2_CFG.WRP, STCU2_ERR_STAT.UFSF and STCU2_ERR_STAT.RFSF. See related register description for additional details

The STCU2 module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register
- Any write operation on Double Security Key register applying a wrong sequence of keys (the two write operations cannot be interleaved with other access to STCU2 registers)
- Any write operation performed on a register when the Double Security keys have not been applied
- Any write operation performed on registers when the STCU2_CFG.WRP bit is set and the access is performed through software (IPS interface)
- Any write operation performed through software (IPS interface) on Read/On-Line_Write/Off-Line registers
- Any write operation performed off-line on registers in which writing operation is allowed only in the On-Line Self-Test phase when STCU2_CFG[WRP] = 1
- Any write operation performed on Read Only registers

The registers of the STCU2 module are accessible (read/write) in each access mode: user, supervisor or test.

In case there are write operations on bits marked as reserved, the transfer error is not generated.

NOTE

See the chip-specific STCU2 information for the wait time necessary to write to the online registers.

75.6 BISTs and BIST partitions

75.6.1 Definition: BIST

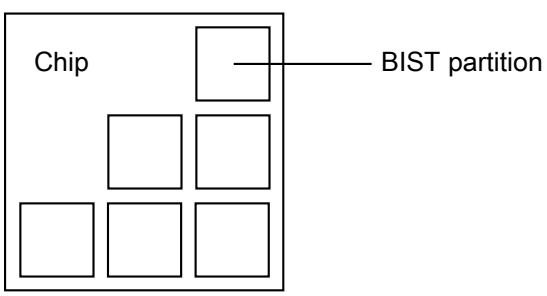
A *built-in self-test (BIST)* is a test that the chip can execute to verify the functional integrity of a part of itself. The chip uses STCU2 and other on-chip hardware to execute a BIST.

75.6.2 Definition: BIST partition

A *built-in self-test (BIST) partition* is a part of a chip for which a BIST has been defined. The hardware that is included in a given BIST partition is chip-specific. For a list of the hardware included in each of the BIST partitions on this chip, see the chip-specific STCU2 information.

75.6.3 Example: BIST partitions on a chip

This is an example of a chip with six built-in self-test (BIST) partitions:



75.6.4 Types of BIST partitions

The STCU2 module on this chip supports the following type or types of built-in self-test (BIST) partitions:

BIST partition type	Description
Memory BIST (MBIST) partition	An SRAM or ROM block

75.7 BIST sequences

75.7.1 Definition: BIST sequence

A *built-in self-test (BIST) sequence* is a programmable series of one or more phases, each of which executes one or more individual BISTs.

75.7.2 Example: Single-phase BIST sequence

This is an example of a single-phase BIST sequence in which STCU2 executes only one BIST:

Phase	BIST executed
0	MBIST 16

75.7.3 Example: Multiphase BIST sequence

This is an example of a multiphase BIST sequence in which STCU2 executes more than one BIST in parallel in some phases:

Phase	BISTs executed
0	MBISTs 7, 38
1	MBIST 23
2	MBISTs 6, 16, 29, 23

75.7.4 Types of BIST sequences

The STCU2 module on this chip supports these types of built-in self-test (BIST) sequences:

BIST sequence type	Description
Online BIST sequence	A BIST sequence that STCU2 executes during runtime at the request of software. Application software configures and initiates execution of the BIST sequence by loading values into STCU2 registers.

Table continues on the next page...

BIST sequence type	Description
	Use this type of BIST sequence when you want to test the chip hardware but need to minimize the system startup time. Also use it for failure diagnostics and quality control within a manufacturing environment. You can choose the most appropriate point during runtime at which to execute this type of BIST sequence.
Offline BIST sequence	A BIST sequence that STCU2 automatically executes during reset, either when power is first applied to the chip or when the chip resets STCU2. The chip's SSCM module configures and initiates execution of the BIST sequence by loading values into STCU2 registers from Device Configuration Format (DCF) records that are stored in the chip's UTEST flash memory. Use this type of BIST sequence when you want to test the chip hardware before runtime. The location of the DCF record area is chip-specific. The size of the DCF record area is sequence-specific. For the location of the DCF record area, see the chip-specific STCU2 information.

75.7.5 Supported BIST sequences

NXP has validated and therefore supports only specific built-in self-test (BIST) sequences for this chip. For a list of the supported BIST sequences, see the chip-specific STCU2 information.

75.7.6 Definition: Default offline BIST sequence

The *default offline built-in self-test (BIST) sequence* is the offline BIST sequence whose configuration NXP has programmed into the chip's Device Configuration Format (DCF) records that are stored in the chip's UTEST flash memory.

If you choose to use the default offline BIST sequence, or if it's the only one supported by NXP, there is no need to create additional DCF records for the offline BIST sequence.

75.8 Functional description

75.8.1 FSM description

The module has three state machines that work together: Master State Machine, Loader/Shifter State Machine, and the Watchdog State Machine. Basically the Master State Machine is the core unit of the STCU2 module. It coordinates all the Self Test operations and the other State Machines. The Loader/Shifter State Machine is used to program the MBIST parameters and to retrieve the related results depending on the parameters stored into the STCU2 registers and under the control of the Master State Machine. The Watchdog State Machine evaluates all the schedule time for MBIST and the time-out in case of wrong STCU2 programming.

75.8.2 Reset management

Assertion of the STCU2 reset input signal initializes the STCU2 module status, forcing the Off-Line Self-Test condition and cleaning up all the registers and state machine of the module.

The reset signals generated by the STCU2 module at the end of the Self-Test execution phase depends on the Self-Test conditions.

75.8.2.1 Off-Line reset generation

At the end of the Off-Line Self-Test execution the STCU2 raises the global functional reset.

75.8.2.2 On-Line reset generation

See the chip-specific STCU2 information for details about On-Line reset generation.

75.8.3 Built-in self-test scheduling

The STCU2 module has been designed to be very flexible allowing to program the parallel/serial execution of the MBIST depending on the power/timing/coverage constraints. The limitation in the programming flexibility is described in Design implementation information.

The mechanism used to provide this flexibility is a linked list where the starting pointer is the bitfield STCU2_CFG.PTR. The first MBIST is mapped to the address 10h, the second to 11h and so on. The additional pointers are in the bitfield STCU2_MB_CTRL[Y].PTR

for MBIST (where Y is the selected MBIST) and have to be filled depending on the selected sequence of run. The additional bitfield STCU2_MB_CTRL.CSM provide the flexibility to run concurrently or sequentially the chosen set of the MBIST or to close the linked list setting the NIL pointer. For more details, see and STCU2 MBIST Control Register (STCU2_MB_CTRL*n*).

75.8.4 ABORT management

The STCU2 module provides On-Line Self-Test execution abort. When the abort is detected, the On-Line Self-Test operation stops running and the status of the currently running MBIST is saved into the related registers to provide intermediate results that might be useful in case of debug or in case of concurrent run.

75.8.4.1 Hardware ABORT management

The STCU2 enters the hardware abort condition as a consequence of a Functional reset which causes Self-Test execution to abort .

When hardware abort is detected the bit ABORTHW of the STCU2_ERR_STAT register is set. This flag allows the software to diagnose what has happened during the On-Line Self-Test execution run.

75.8.5 PLL interface

The STCU2 module is able to directly control the PLL during the Off-Line Self-Test operations depending on the status of the STCU2_RUN[MBPLLEN].

When STCU2_RUN[RUN] is set to start the Off-Line Self-test operations, the STCU2 takes control of the PLL and changes the PLL clock configurations parameters according to the values programmed into the STCU2_PLL_CFG register.

When the STCU2_RUN[MBPLLEN] is active and the MBIST is currently selected, the STCU2 waits for the PLL lock signal. As soon as the lock happens, the STCU2 forces the clock to switch to the PLL output clock. This allows the MBIST to proceed at the PLL output frequency. At the end of the MBIST run the clock source is reverted to the original one.

75.8.6 FCCU interface

The FCCU interface is the hardware flag mechanism towards the system, to indicate the occurrence of an Unrecoverable fault and/or a Recoverable fault failure during the Self Test sequence.

To diagnose physical defects on the two fault signals, a fault injection mechanism is also provided. In this case, the FCCU interface allows the user application to check the integrity of the UF and RF connection lines between the STCU2 and the FCCU. Refer to the description of FCCU fault injection mechanism to understand how the UF/RF set/clear mechanism works.

75.8.7 Watchdogs

The STCU2 implements three different watchdogs to ensure that operations are finished in time.

75.8.7.1 Initialization watchdog timer

The STCU2 has two initialization operating modes: program the self-test sequence and run it (Safety mode) or bypass completely the self-test sequence setting the bit STCU2_CFG.BYP. In case there are faults during the initialization phase preventing the selection of one of these two operating modes, a hard-coded watchdog time-out flags the wrong behavior.

75.8.7.2 Built-in self-test watchdog timer

The MBIST execution time has to be configured as described in STCU2 Watchdog Granularity (STCU2_WDG), to account for the overall execution time of the Self-Test sequence. In case the selected MBISTs are not yet completed during the Off-line Self-Test assigned time, the current or MBISTs execution is interrupted and a failure is flagged into STCU2_ERR_STAT.WDTO and STCU2_MBEL/M/H registers while in case of On-Line Self-Test into STCU2_ERR_STAT.WDTOSW and STCU2_MBELSW/MSW/HSW.

In case of multiple sequential run in the same On-Line/Off-Line session and the time-out happens in the middle of a sequential run, the next sequential run will be skipped and the execution ends with the current updated status of the registers reported above.

In all the cases, the STCU2 reset generation is managed in one of the modes described in Reset management.

75.8.7.3 Register write-access watchdog timer

As explained in the STCU2_SKC register description:

- A key mechanism protects STCU2 registers during the self-test configuration phase (both online/offline) by preventing any unwanted access.
- A hardware watchdog timer locks register-write access after a number of STCU2 clock cycles. To refresh the hardware watchdog timer before it times out, write only security key 2 to STCU2_SKC.

AUTOLOCK_VALUE is the number of STCU2 clock cycles after which the hardware watchdog timer locks register-write access. See the chip-specific STCU2 information for the value of AUTOLOCK_VALUE.

NOTE

For write completion, each DCF record requires a number of STCU2 clock cycles equal to DCF_COMPLETION. The maximum number of DCF records that can be executed before the watchdog timer must be refreshed is AUTOLOCK_VALUE divided by DCF_COMPLETION. See the chip-specific STCU2 information for the value of DCF_COMPLETION.

The hardware watchdog timer is particularly useful in case STCU2_CFG[WRP] is 0 during the software self-test configuration. In this case, the software application might enable write access to the STCU2 registers.

75.8.8 SSCM interface

The SSCM interface is used to program the STCU2's configuration parameters without the CPU intervention after a reset trigger event initializes the STCU2. This bus interface has the priority over the IPS.

75.9 Register description

During Offline SELFTEST, once the STCU2 registers are configured they cannot be overridden.

During Online SELFTEST, once the STCU2 registers are configured they cannot be overridden via IPS till the selftest is complete.

The STCU2 registers are listed in this section.

STCU2 memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	STCU2 Run Register (STCU2_RUN)	32	R/W	0000_0000h	75.9.1/3223
4	STCU2 Run Software Register (STCU2_RUNSW)	32	R/W	0000_0000h	75.9.2/3224
8	STCU2 SK Code Register (STCU2_SKC)	32	W	0000_0000h	75.9.3/3225
C	STCU2 Configuration Register (STCU2_CFG)	32	R/W	See section	75.9.4/3226
14	STCU2 Watchdog Granularity (STCU2_WDG)	32	R/W	See section	75.9.5/3229
24	STCU2 Error Register (STCU2_ERR_STAT)	32	R/W	0000_0000h	75.9.6/3230
28	STCU2 Error FM Register (STCU2_ERR_FM)	32	R/W	See section	75.9.7/3233
44	STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL)	32	R	0000_0000h	75.9.8/3235
48	STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)	32	R	0000_0000h	75.9.9/3240
4C	STCU2 Off-Line MBIST Status High Register (STCU2_MBSH)	32	R	0000_0000h	75.9.10/ 3242
50	STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL)	32	R	0000_0000h	75.9.11/ 3243
54	STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM)	32	R	0000_0000h	75.9.12/ 3246
58	STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH)	32	R	0000_0000h	75.9.13/ 3248
5C	STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW)	32	R	0000_0000h	75.9.14/ 3249
60	STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW)	32	R	0000_0000h	75.9.15/ 3252
64	STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW)	32	R	0000_0000h	75.9.16/ 3254
68	STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW)	32	R	0000_0000h	75.9.17/ 3255
6C	STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW)	32	R	0000_0000h	75.9.18/ 3259
70	STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW)	32	R	0000_0000h	75.9.19/ 3261
74	STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)	32	R/W	See section	75.9.20/ 3261
78	STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)	32	R/W	See section	75.9.21/ 3265
7C	STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)	32	R/W	See section	75.9.22/ 3267

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
600	STCU2 MBIST Control Register (STCU2_MB_CTRL0)	32	R/W	See section	75.9.23/ 3268
604	STCU2 MBIST Control Register (STCU2_MB_CTRL1)	32	R/W	See section	75.9.23/ 3268
608	STCU2 MBIST Control Register (STCU2_MB_CTRL2)	32	R/W	See section	75.9.23/ 3268
60C	STCU2 MBIST Control Register (STCU2_MB_CTRL3)	32	R/W	See section	75.9.23/ 3268
610	STCU2 MBIST Control Register (STCU2_MB_CTRL4)	32	R/W	See section	75.9.23/ 3268
614	STCU2 MBIST Control Register (STCU2_MB_CTRL5)	32	R/W	See section	75.9.23/ 3268
618	STCU2 MBIST Control Register (STCU2_MB_CTRL6)	32	R/W	See section	75.9.23/ 3268
61C	STCU2 MBIST Control Register (STCU2_MB_CTRL7)	32	R/W	See section	75.9.23/ 3268
620	STCU2 MBIST Control Register (STCU2_MB_CTRL8)	32	R/W	See section	75.9.23/ 3268
624	STCU2 MBIST Control Register (STCU2_MB_CTRL9)	32	R/W	See section	75.9.23/ 3268
628	STCU2 MBIST Control Register (STCU2_MB_CTRL10)	32	R/W	See section	75.9.23/ 3268
62C	STCU2 MBIST Control Register (STCU2_MB_CTRL11)	32	R/W	See section	75.9.23/ 3268
630	STCU2 MBIST Control Register (STCU2_MB_CTRL12)	32	R/W	See section	75.9.23/ 3268
634	STCU2 MBIST Control Register (STCU2_MB_CTRL13)	32	R/W	See section	75.9.23/ 3268
638	STCU2 MBIST Control Register (STCU2_MB_CTRL14)	32	R/W	See section	75.9.23/ 3268
63C	STCU2 MBIST Control Register (STCU2_MB_CTRL15)	32	R/W	See section	75.9.23/ 3268
640	STCU2 MBIST Control Register (STCU2_MB_CTRL16)	32	R/W	See section	75.9.23/ 3268
644	STCU2 MBIST Control Register (STCU2_MB_CTRL17)	32	R/W	See section	75.9.23/ 3268
648	STCU2 MBIST Control Register (STCU2_MB_CTRL18)	32	R/W	See section	75.9.23/ 3268
64C	STCU2 MBIST Control Register (STCU2_MB_CTRL19)	32	R/W	See section	75.9.23/ 3268
650	STCU2 MBIST Control Register (STCU2_MB_CTRL20)	32	R/W	See section	75.9.23/ 3268
654	STCU2 MBIST Control Register (STCU2_MB_CTRL21)	32	R/W	See section	75.9.23/ 3268

Table continues on the next page...

STCU2 memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
658	STCU2 MBIST Control Register (STCU2_MB_CTRL22)	32	R/W	See section	75.9.23/ 3268
65C	STCU2 MBIST Control Register (STCU2_MB_CTRL23)	32	R/W	See section	75.9.23/ 3268
660	STCU2 MBIST Control Register (STCU2_MB_CTRL24)	32	R/W	See section	75.9.23/ 3268
664	STCU2 MBIST Control Register (STCU2_MB_CTRL25)	32	R/W	See section	75.9.23/ 3268
668	STCU2 MBIST Control Register (STCU2_MB_CTRL26)	32	R/W	See section	75.9.23/ 3268
66C	STCU2 MBIST Control Register (STCU2_MB_CTRL27)	32	R/W	See section	75.9.23/ 3268
670	STCU2 MBIST Control Register (STCU2_MB_CTRL28)	32	R/W	See section	75.9.23/ 3268
674	STCU2 MBIST Control Register (STCU2_MB_CTRL29)	32	R/W	See section	75.9.23/ 3268
678	STCU2 MBIST Control Register (STCU2_MB_CTRL30)	32	R/W	See section	75.9.23/ 3268
67C	STCU2 MBIST Control Register (STCU2_MB_CTRL31)	32	R/W	See section	75.9.23/ 3268
680	STCU2 MBIST Control Register (STCU2_MB_CTRL32)	32	R/W	See section	75.9.23/ 3268
684	STCU2 MBIST Control Register (STCU2_MB_CTRL33)	32	R/W	See section	75.9.23/ 3268
688	STCU2 MBIST Control Register (STCU2_MB_CTRL34)	32	R/W	See section	75.9.23/ 3268
68C	STCU2 MBIST Control Register (STCU2_MB_CTRL35)	32	R/W	See section	75.9.23/ 3268
690	STCU2 MBIST Control Register (STCU2_MB_CTRL36)	32	R/W	See section	75.9.23/ 3268
694	STCU2 MBIST Control Register (STCU2_MB_CTRL37)	32	R/W	See section	75.9.23/ 3268
698	STCU2 MBIST Control Register (STCU2_MB_CTRL38)	32	R/W	See section	75.9.23/ 3268
69C	STCU2 MBIST Control Register (STCU2_MB_CTRL39)	32	R/W	See section	75.9.23/ 3268
6A0	STCU2 MBIST Control Register (STCU2_MB_CTRL40)	32	R/W	See section	75.9.23/ 3268
6A4	STCU2 MBIST Control Register (STCU2_MB_CTRL41)	32	R/W	See section	75.9.23/ 3268
6A8	STCU2 MBIST Control Register (STCU2_MB_CTRL42)	32	R/W	See section	75.9.23/ 3268
6AC	STCU2 MBIST Control Register (STCU2_MB_CTRL43)	32	R/W	See section	75.9.23/ 3268

Table continues on the next page...

STCU2 memory map (continued)

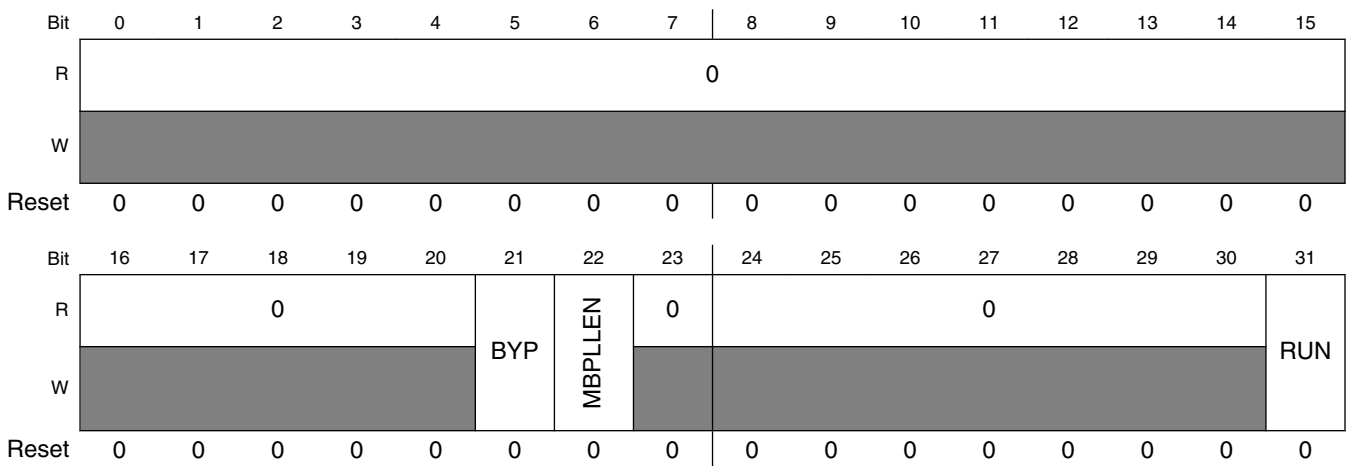
Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
6B0	STCU2 MBIST Control Register (STCU2_MB_CTRL44)	32	R/W	See section	75.9.23/ 3268

75.9.1 STCU2 Run Register (STCU2_RUN)

The STCU2_RUN register defines the RUN bit to start the off-line self testing procedure.

The R/W fields in this register are readable at any time. However, you can write to these fields only when Off-line Self-Test phase is still active.

Address: 0h base + 0h offset = 0h



STCU2_RUN field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 BYP	Bypass mode 0 Off-Line Self-Test is active. In case the RUN bit into the STCU2_RUN register is not set before the Hard-Coded WDG Time-out, the WDTO error is generated into the STCU2_ERR_STAT register. 1 Off-Line Self-Test is completely bypassed and the access to STCU2 is locked until the STCU2_SKC register is written according to the request access mode.
22 MBPLLEN	Off-Line MBIST with PLL Enabled 0 Off-Line MBIST is executed without using the on-chip PLL. 1 Off-Line MBIST is executed enabling the on-chip PLL control interface selecting the parameters defined into STCU2_PLL_CFG register.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

STCU2_RUN field descriptions (continued)

Field	Description
24–30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RUN	RUN: RUN The RUN bit is automatically cleared by the STCU2 when the off-line self testing procedure has been completed. 0 Idle 1 Off-Line Self testing procedure is running

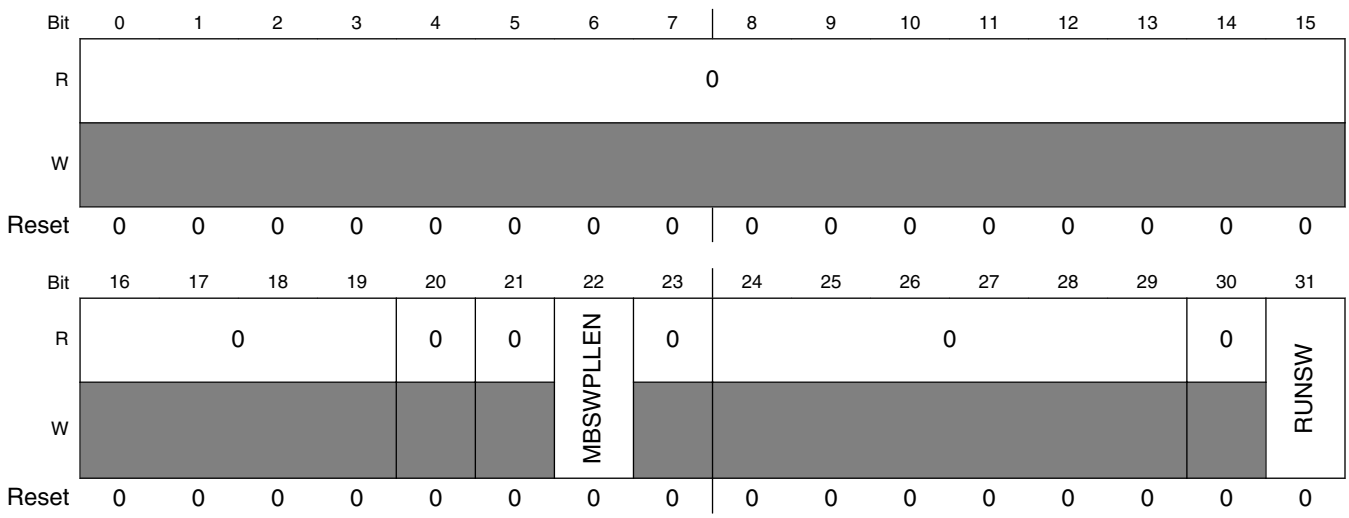
75.9.2 STCU2 Run Software Register (STCU2_RUNSW)

The STCU2_RUNSW register defines the RUN bit to start the on-line self testing procedure.

The R/W fields in this register are readable at any time. However, you can write to these fields only when both of the following conditions are true:

- STCU2_CFG[WRP] = 0
- On-line Self-Test phase is active. The On-Line condition starts when the Off-Line is over. In this condition, system is alive and the SW can program the STCU2

Address: 0h base + 4h offset = 4h



STCU2_RUNSW field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

STCU2_RUNSW field descriptions (continued)

Field	Description
20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 MBSWPLLEN	On-Line MBIST with PLL Enabled 0 On-Line MBIST is executed without using the on-chip PLL. 1 On-Line MBIST is executed using the PLL configuration provided by software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 RUNSW	RUNSW The RUNSW bit is automatically cleared by the STCU2 when the on-line self testing procedure has been completed. 0 Idle 1 On-line self testing procedure is running

75.9.3 STCU2 SK Code Register (STCU2_SKC)

The STCU2_SKC register implements the security key code mechanism needed to access in write mode to the other STCU2 registers. In order to unlock the STCU2 access after:

- The STCU2 asynchronous reset
- The end of the STCU2 run

the software (IPS bus)or the SSCM interfaces have to apply the following sequence:

- write the key1 into the STCU2_SKC register
- write the key2 into the STCU2_SKC register

Depending on the off/on-line test step, the two keys will be different. Byte write operation is not allowed since the full key has to be recognized as one unit.

It should be noted that after the Self-Test sequence has been completed or the Bypass feature has been enabled (setting the bit BYP into the STCU2_CFG), the SSCM interface is no longer available.

In case of invalid access or sequence (Key1/2 have to be applied consecutively), a transfer error on the IPS or SSCM busis asserted depending on the selected source. The STCU2 write access is locked and to unlock the access the sequence has to be applied again.

In case the STCU2 register access lasts more cycles than the one defined into the Hard-coded WDG time-out, the STCU2 write access is locked and the WDG and Register ITF clocks are switched off. Also in this case, in order to enable again the write access to the STCU2 and the WDG and Register ITF clocks, it is required to apply again the sequence.

In case it is required to extend the STCU2 register access cycles before the Hard-coded WDG time-out expires, only the Key2 has to be applied. The effect of this write operation is to re-initialize the WDG time-out counter. In such a condition, Key1 has not to be applied otherwise a transfer error on the IPS bus is asserted depending on the selected source. The STCU2 write access is locked and to unlock the access the sequence has to be applied again.

The STCU2_SKC register is not readable. The value 00000000h is always returned in case of read operation.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	SKC																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_SKC field descriptions

Field	Description
0–31 SKC	STCU2 security key code for off-line Test = D3FEA98Bh: Key1 to unlock the write access the STCU2 = 2C015674h: Key2 to unlock the write access the STCU2 STCU2 security key code for on-line Test = 753F924Eh: Key1 to unlock the write access the STCU2 = 8AC06DB1h: Key2 to unlock the write access the STCU2

75.9.4 STCU2 Configuration Register (STCU2_CFG)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_CFG register includes the global configuration of the STCU2 and can be updated both in the Off/On-Line Test steps.

The access to this register is described in the following figure. It further depends on the state of the WRP bit as follows:

- When WRP = 0: The register can be written during the on-line Self-Test case without restrictions
- When WRP = 1:
  - The only bit that can be written without any restriction is WRP.
  - In case there are software operations that write all the register's bits, a transfer error is raised only if the value of the selected byte written differs from the current status of the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate this single bit clean capability.

MBIST can be run in three different test types according to how the STCU2_CFG register bits 27 and 28 are set.

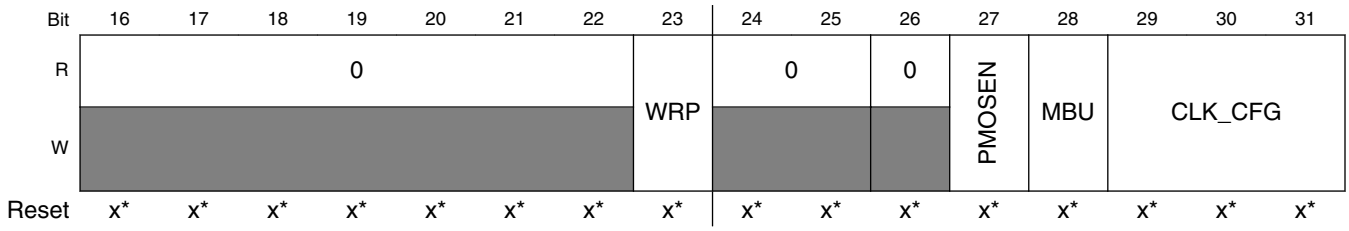
The following table describes the test algorithm and coverage of the different test types.

Table 75-4. MBIST algorithm and coverage according to test type

MBIST Type	STCU2_CFG[PMOSEN]	STCU2_CFG[MBU]	MBIST Algorithm	Internal Nodes/ Circuits Additionally Covered by Test	Use Case	Test Cycle
Full Test	1	0	Test memory using all built in algorithms. Open PMOS algorithm included	Address Decoder and Bitcell as well as resistive defects in the address decoder	Used by MCU production test. Recommended to use in the field for MBIST online self-test.	About 3 to 4x more than Auto Test
Reduced Test	0	0	Test memory using all built in algorithms except the open PMOS algorithm.	Address Decoder and Bitcell	Recommended to use in the field for MBIST online self-test if the Full Test takes too long.	About 10% less than Full Test
Auto Test	x	1	A smaller set of algorithms which target latent defect mechanisms.	Latent defects such as NBTI of the PMOS transistors in the bitcells	Recommended to use for MBIST offline self-test as an optimum balance of test time vs. fault coverage.	About 3 to 4x faster than Full Test

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	PTR														
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*



* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_CFG field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1–7 PTR	MBIST pointer PTR defines the logical pointer to the first MBIST to be scheduled when the self testing procedure is enabled. PTR is the entry pointer to a linked list of BIST descriptors. You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0. 10h to (10h + MBIST -1): pointer to MBIST 7Fh: pointer to NIL. No BIST execution.
16–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23 WRP	Write Protection 0: Specific STCU2 registers can be written through IPS bus interface after Off-Line Self-Test sequence has been completed 1: Specific STCU2 registers cannot be written though IPS also after Off-Line Self-Test sequence has been completed preventing in such a way any user application write operation
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 PMOSEN	MBIST PMOS Test Enable When MBU is 1 then "Auto test" MBIST test type will be selected regardless of the setting of this field. 0 When MBU is 0 then selects the "Reduced test" MBIST test type. 1 When MBU is 0 then selects the "Full test" MBIST test type.
28 MBU	MBIST MBU Test Enabled 0 Either "Full test" or "Reduced test" MBIST test type will be selected according to the PMOSEN value. 1 Selects "Auto test" MBIST test type regardless of the PMOSEN value.
29–31 CLK_CFG	Logic, Memory BIST and STCU2 core CLK Clock configuration CLK_CFG defines the ratio between the sys_clk and the TCK used to program the MBIST and the STCU2 core clock. The punch-out mechanism is used to generate the derived clocks. The allowed configurations are the following:

Table continues on the next page...

STCU2_CFG field descriptions (continued)

Field	Description
	000: sys_clk 001: sys_clk/2 010: sys_clk/3 ... 101: sys_clk/6 110: sys_clk/7 111: sys_clk/8 You can always read this field. In the off-line Self-Test phase, you can always write to this field. In the on-line Self-Test phase, you can write to this field only when WRP = 0.

75.9.5 STCU2 Watchdog Granularity (STCU2_WDG)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_WDG register defines the time limit for MBIST execution, providing a protection mechanism in case of deadlock or endless conditions during the self test procedure.

When the offline self test sequence is not run and the STCU2_CFG[BYP] field is not set, bits 15..0 define the time-out before the STCU2_ERR[WDTO] field is set and the STCU2 core clock is switched off.

In case offline/online self test sequence is run, the STCU2_WDG value defines the time-out of the execution run.

The register fields are always readable. You can write to these fields when either of the following conditions is true:

- Offline self test phase is active
- Online self test phase is active and CFG[WRP] = 0

Address: 0h base + 14h offset = 14h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	WDGEOC																															
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

STCU2_WDG field descriptions

Field	Description
0–31 WDGEOC	Watchdog End of Count Timer This value defines the time limit for offline and online self test execution to ensure correct operation within the allowed time. **NOTE:** If this register is programmed while STCU2_RUN[RUN]=1, the new STCU2_WDG value, programmed after STCU2_RUN, does not take effect. The allowed values are: 0000 0000h: 1 × 16 STCU2 Core clock cycles 0000 0001h: 2 × 16 STCU2 Core clock cycles 0000 0002h: 3 × 16 STCU2 Core clock cycles ... FFFF FFFDh: 4294967294 × 16 STCU2 Core clock cycles FFFF FFFEh: 4294967295 × 16 STCU2 Core clock cycles FFFF FFFFh: 4294967296 × 16 STCU2 Core clock cycles

75.9.6 STCU2 Error Register (STCU2_ERR_STAT)

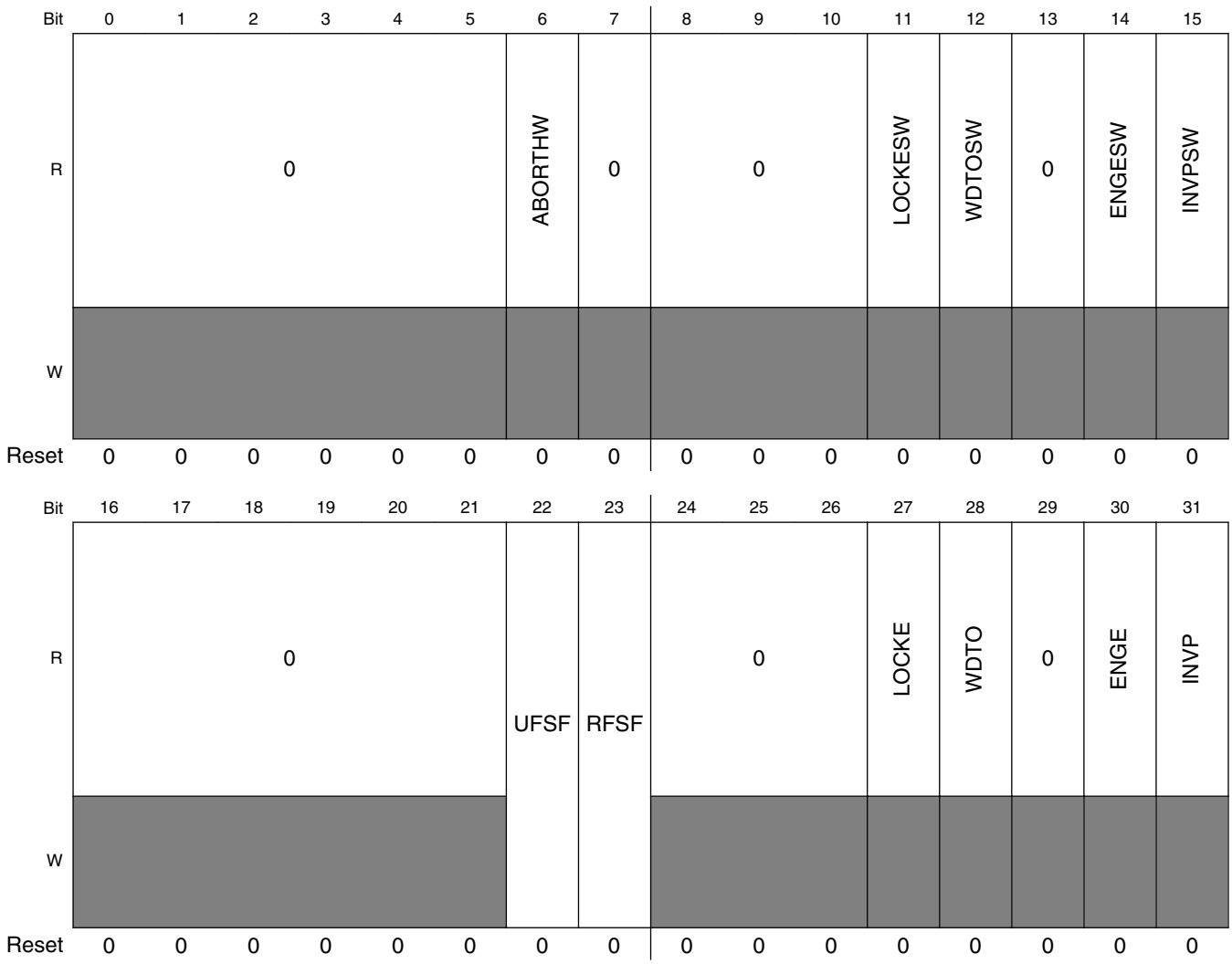
The STCU2_ERR_STAT register includes the status flags related to the STCU2 internal fault conditions occurred during the configuration or the On/Off-Line self testing execution.

The UFSF and RFSF can be set/cleared using the FCCU dedicated channels.

The access to this register is described in the following figure and as follows:

- If you select the byte write capability to write only the UFSF and RFSF, then there is no restriction in writing these bits.
- If your software performs the write operations on other bits besides UFSF/RFSF, then a transfer error is generated only if the value you are writing to the other bits differs from their value currently stored in the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate the UFSF/RFSF single bit set/clean capability.

Address: 0h base + 24h offset = 24h



STCU2_ERR_STAT field descriptions

Field	Description
0–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 ABORTHW	On-line hardware abort flag You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 No hardware abort was requested during the On-Life Self-Test sequence 1 A hardware abort was detected during the On-Life Self-Test sequence
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 LOCKESW	On-Line LOCK Error

Table continues on the next page...

STCU2_ERR_STAT field descriptions (continued)

Field	Description
	You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 In case PLL is enabled, it is correctly locked during the Self-Test sequence 1 When the PLL is enabled, this flag highlight that there has been an unexpected PLL unlock event during the On-Line Self-Test sequence execution. The On-Line Self-test run is stopped and the status of the current running MBISTs is saved into the related registers.
12 WDTOSW	On-Line Watchdog time-out You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 MBIST time slot have been completed within the assigned watchdog time. 1 MBIST time slot haven't been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The conditions that flag the failures are the following: • MBIST BEND status flags that at least one of the selected MBIST run is not finished
13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14 ENGESW	On-Line Engine Error You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: • STCU2 state machines (Watchdog, Master and Loader/Shifter) selects an unused code • Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned • Serial bend status of the selected MBIST and the global BEND flag are not aligned
15 INVPSW	On-Line Invalid Pointer You can always read this field. The content of this field is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1. 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: • Initial MBIST pointer is out of range • Error in the MBIST linking (execution generates an infinite loop)
16–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 UFSF	Unrecoverable Faults Status Flag This flag reports the global status of the UF. This field can be set or cleared using the FCCU dedicated channel, and can also be set or cleared by software. 0 No errors that trigger the Unrecoverable Faults condition 1 There are errors that trigger the Unrecoverable Faults condition
23 RFSF	Recoverable Faults Status Flag This flag reports the global status of the RF. This field can be set or cleared using the FCCU dedicated channel, and can also be set or cleared by software. 0 No errors that trigger the Recoverable Faults condition 1 There are errors that trigger the Recoverable Faults condition

Table continues on the next page...

STCU2_ERR_STAT field descriptions (continued)

Field	Description
24–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LOCKE	Off-Line LOCK Error 0 The PLL is correctly locked during the Self-Test sequence 1 When the PLL is enabled, this flag highlights that there has been an unexpected PLL unlock event during the Off-Line Self-Test sequence execution. The Off-Line Self-test run is stopped and the status of the current running MBISTs is saved into the related registers. The LOCK signal is monitored during the the MBIST run when STCU2_RUN[MBPLLEN] = 1.
28 WDTO	Off-Line Watchdog time-out 0 MBIST time slot have been completed within the assigned watchdog time. 1 MBIST time slot haven't been completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. The conditions that flag the failures are the following: • The STCU2_RUN[RUN] bit or the STCU2_RUN[BYP] bit is not set before the Watchdog reaches the End of Count • MBIST BEND status flags that at least one of the selected MBIST run is not finished
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 ENGE	Off-Line Engine Error 0 Valid Engine execution 1 Invalid Engine execution. The following conditions set this bit: • STCU2 state machines (Watchdog, Master and Loader/Shifter) selects an unused code • Serial pass/fail status of the selected MBIST and the global BBAD flag are not aligned • Serial bend status of the selected MBIST and the global BEND flag are not aligned
31 INVP	Off-Line Invalid pointer 0 Valid linked pointer list 1 Invalid linked pointer list. The following conditions set this bit: • Initial MBIST pointer is out of range • Error in the MBIST linking (execution generates an infinite loop)

75.9.7 STCU2 Error FM Register (STCU2_ERR_FM)

NOTE

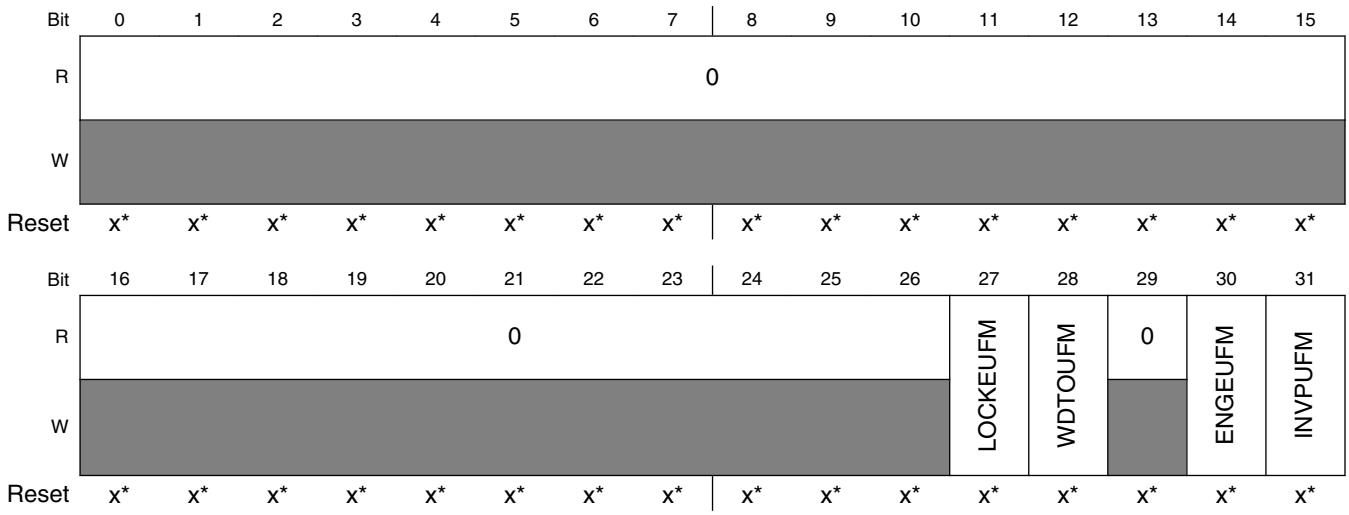
The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_ERR_FM register defines the fault mapping of the STCU2 faults described into the register STCU2_ERR_STAT in terms of Unrecoverable or Recoverable Fault. All sources of internal faults can be routed to UF and RF.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 28h offset = 28h



* Notes:

- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_ERR_FM field descriptions

Field	Description
0–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27 LOCKEUFM	PLL LOCK Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
28 WDTOUFM	Watchdog time-out Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30 ENGEUFM	Engine Error Unrecoverable fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Fault Mapping
31 INVPUFM	Invalid Pointer Unrecoverable Fault Mapping 0 Recoverable Fault Mapping 1 Unrecoverable Mapping

75.9.8 STCU2 Off-Line MBIST Status Low Register (STCU2_MBSL)

The STCU2_MBSL register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs .

Address: 0h base + 44h offset = 44h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBS31	MBS30	MBS29	MBS28	MBS27	MBS26	MBS25	MBS24	MBS23	MBS22	MBS21	MBS20	MBS19	MBS18	MBS17	MBS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBS15	MBS14	MBS13	MBS12	MBS11	MBS10	MBS9	MBS8	MBS7	MBS6	MBS5	MBS4	MBS3	MBS2	MBS1	MBS0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSL field descriptions

Field	Description
0 MBS31	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBS30	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished.

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBS29	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
3 MBS28	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBS27	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBS26	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBS25	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBS24	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBS23	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBS22	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBS21	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBS20	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBS19	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBS18	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBS17	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBS16	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBS15	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

Field	Description
17 MBS14	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBS13	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBS12	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBS11	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBS10	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBS9	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBS8	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBS7	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished.

Table continues on the next page...

STCU2_MBSL field descriptions (continued)

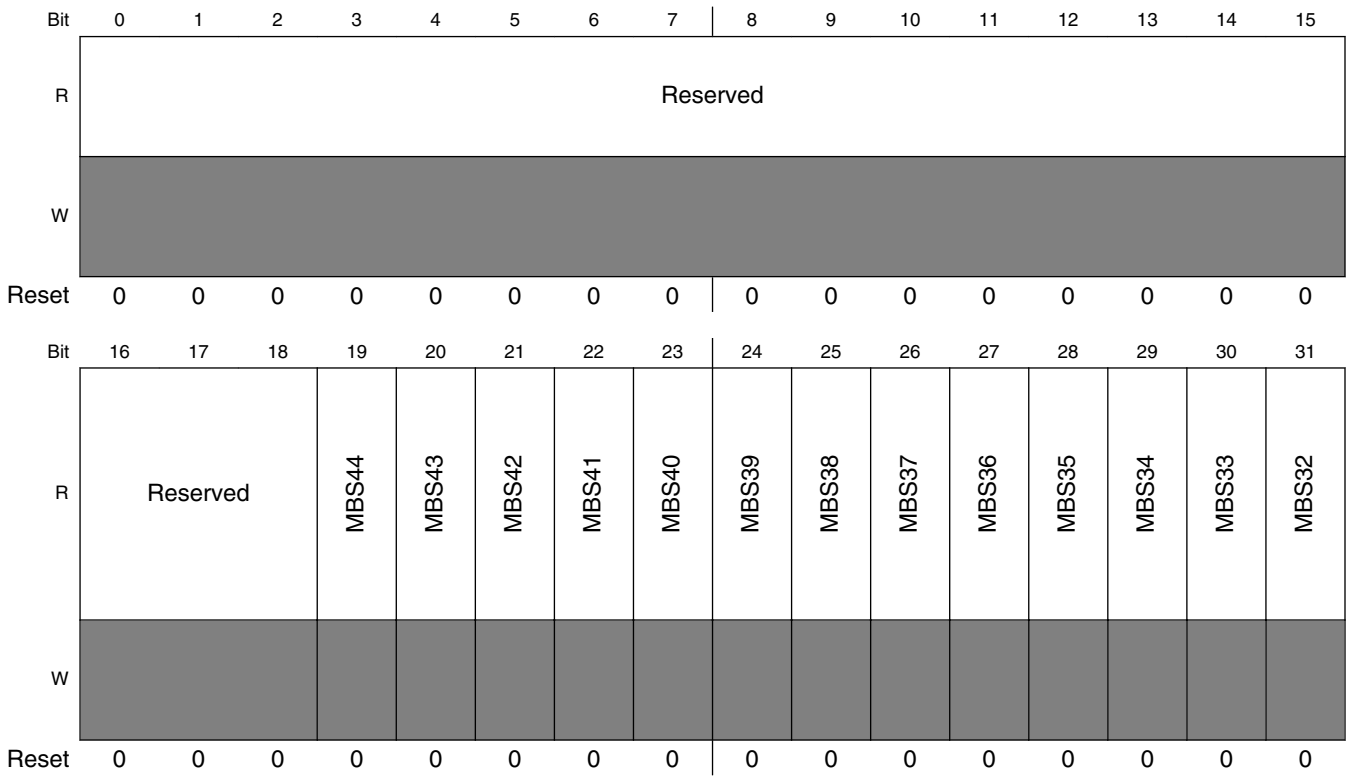
Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBS6	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBS5	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBS4	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBS3	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBS2	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBS1	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBS0	Off-Line status (NMCUT range = 0 to 31) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEL register reports the MBIST is finished. 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

75.9.9 STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM)

The STCU2_MBSM register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs .

Address: 0h base + 48h offset = 48h



STCU2_MBSM field descriptions

Field	Description
0–18 Reserved	This field is reserved.
19 MBS44	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBS43	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished).

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBS42	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBS41	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBS40	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBS39	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBS38	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBS37	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBS36	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBS35	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBSM field descriptions (continued)

Field	Description
	NOTE: This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBS34	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBS33	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBS32	MBSx: Off-Line status (NMCUT range = 32 to 63) of the selected MBIST **NOTE:** This bit is meaningful when the related bit of the STCU2_MBEM register reports the MBIST is finished). 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

75.9.10 STCU2 Off-Line MBIST Status High Register (STCU2_MBSH)

The STCU2_MBSH register includes the results corresponding to the execution of the selected Off-Line MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 4Ch offset = 4Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSH field descriptions

Field	Description
0–31 Reserved	This field is reserved.

75.9.11 STCU2 Off-Line MBIST End Flag Low Register (STCU2_MBEL)

The STCU2_MBEL register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 50h offset = 50h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBE31	MBE30	MBE29	MBE28	MBE27	MBE26	MBE25	MBE24	MBE23	MBE22	MBE21	MBE20	MBE19	MBE18	MBE17	MBE16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBE15	MBE14	MBE13	MBE12	MBE11	MBE10	MBE9	MBE8	MBE7	MBE6	MBE5	MBE4	MBE3	MBE2	MBE1	MBE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEL field descriptions

Field	Description
0 MBE31	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBE30	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEL field descriptions (continued)

Field	Description
2 MBE29	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBE28	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
4 MBE27	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBE26	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBE25	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBE24	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
8 MBE23	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBE22	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBE21	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBE20	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBE19	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBE18	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEL field descriptions (continued)

Field	Description
14 MBE17	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBE16	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBE15	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBE14	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBE13	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBE12	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBE11	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBE10	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBE9	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBE8	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBE7	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBE6	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEL field descriptions (continued)

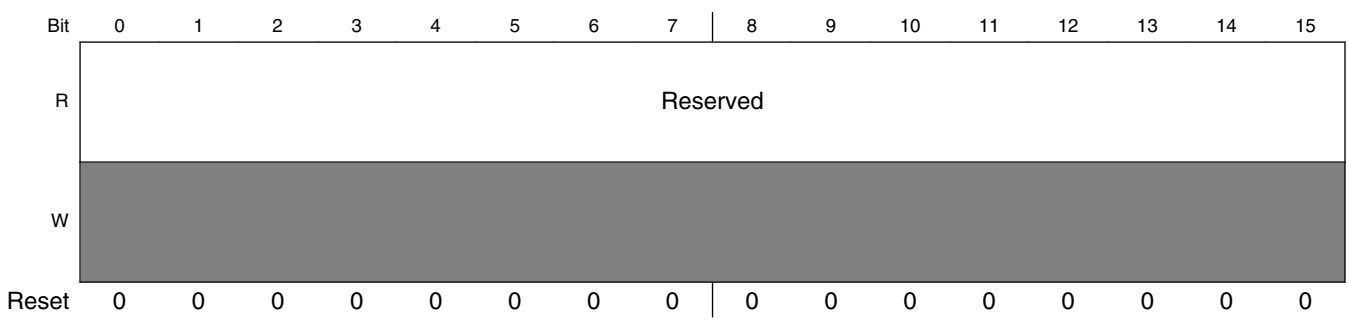
Field	Description
26 MBE5	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBE4	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBE3	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBE2	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBE1	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBE0	Off-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

75.9.12 STCU2 Off-Line MBIST End Flag Medium Register (STCU2_MBEM)

The STCU2_MBEM register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 54h offset = 54h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved			MBE44	MBE43	MBE42	MBE41	MBE40	MBE39	MBE38	MBE37	MBE36	MBE35	MBE34	MBE33	MBE32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEM field descriptions

Field	Description
0–18 Reserved	This field is reserved.
19 MBE44	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
20 MBE43	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBE42	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBE41	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBE40	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBE39	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBE38	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBE37	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEM field descriptions (continued)

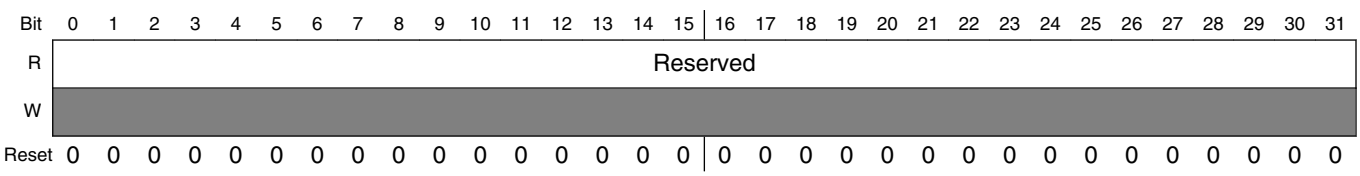
Field	Description
27 MBE36	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBE35	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBE34	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBE33	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBE32	Off-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

75.9.13 STCU2 Off-Line MBIST End Flag High Register (STCU2_MBEH)

The STCU2_MBEH register includes the End Flag related to the execution of the selected Off-Line MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

Address: 0h base + 58h offset = 58h



STCU2_MBEH field descriptions

Field	Description
0–31 Reserved	This field is reserved.

75.9.14 STCU2 On-Line MBIST Status Low Register (STCU2_MBSLSW)

The STCU2_MBSLSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 5Ch offset = 5Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MBSSW31	MBSSW30	MBSSW29	MBSSW28	MBSSW27	MBSSW26	MBSSW25	MBSSW24	MBSSW23	MBSSW22	MBSSW21	MBSSW20	MBSSW19	MBSSW18	MBSSW17	MBSSW16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBSSW15	MBSSW14	MBSSW13	MBSSW12	MBSSW11	MBSSW10	MBSSW9	MBSSW8	MBSSW7	MBSSW6	MBSSW5	MBSSW4	MBSSW3	MBSSW2	MBSSW1	MBSSW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBSLSW field descriptions

Field	Description
0 MBSSW31	On-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
1 MBSSW30	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
2 MBSSW29	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
3 MBSSW28	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
4 MBSSW27	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
5 MBSSW26	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
6 MBSSW25	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
7 MBSSW24	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
8 MBSSW23	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
9 MBSSW22	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
10 MBSSW21	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
11 MBSSW20	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
12 MBSSW19	On-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
13 MBSSW18	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
14 MBSSW17	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
15 MBSSW16	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
16 MBSSW15	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
17 MBSSW14	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
18 MBSSW13	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
19 MBSSW12	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBSSW11	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBSSW10	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBSSW9	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBSSW8	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBSSW7	On-Line status (NMCUT range = 0 to 31) of the selected MBIST

Table continues on the next page...

STCU2_MBSLSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBSSW6	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBSSW5	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBSSW4	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBSSW3	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBSSW2	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBSSW1	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBSSW0	On-Line status (NMCUT range = 0 to 31) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

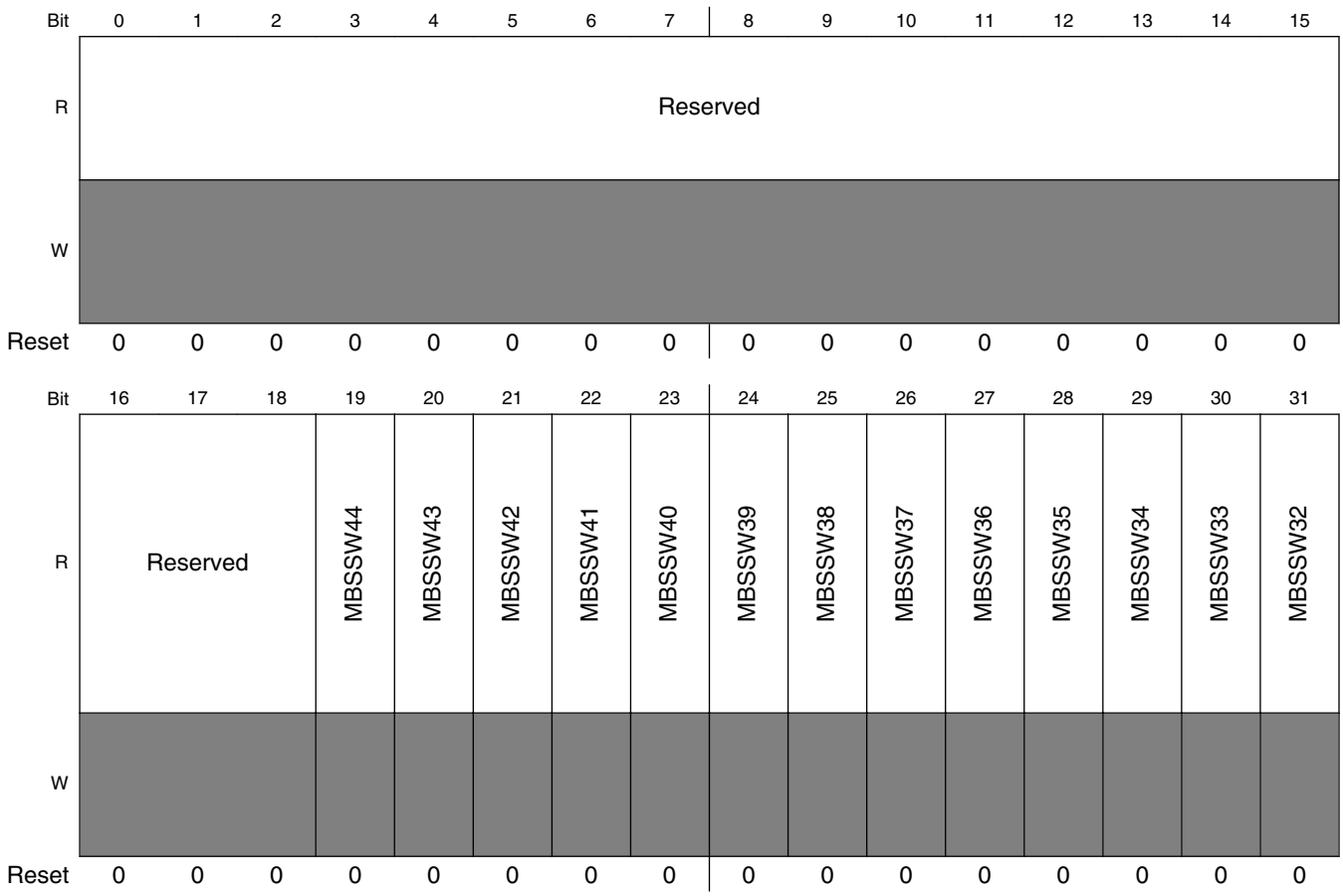
75.9.15 STCU2 On-Line MBIST Status Medium Register (STCU2_MBSMSW)

The STCU2_MBSMSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 32..63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 60h offset = 60h



STCU2_MBSMSW field descriptions

Field	Description
0–18 Reserved	This field is reserved.
19 MBSSW44	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
20 MBSSW43	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
21 MBSSW42	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
22 MBSSW41	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
23 MBSSW40	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST

Table continues on the next page...

STCU2_MBSMSW field descriptions (continued)

Field	Description
	0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
24 MBSSW39	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
25 MBSSW38	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
26 MBSSW37	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
27 MBSSW36	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
28 MBSSW35	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
29 MBSSW34	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
30 MBSSW33	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution
31 MBSSW32	MBSSWx: On-Line status (NMCUT range = 32 to 63) of the selected MBIST 0 Failed NMCUT BIST execution 1 No Fault detected during the NMCUT BIST execution

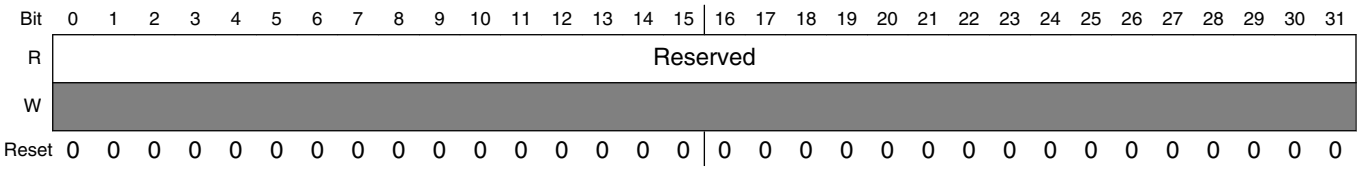
75.9.16 STCU2 On-Line MBIST Status High Register (STCU2_MBSHSW)

The STCU2_MBSHSW register includes the results corresponding to the execution of the selected On-Line MBIST in the range NMCUT = 64..95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 64h offset = 64h



STCU2_MBSHSW field descriptions

Field	Description
0–31 Reserved	This field is reserved.

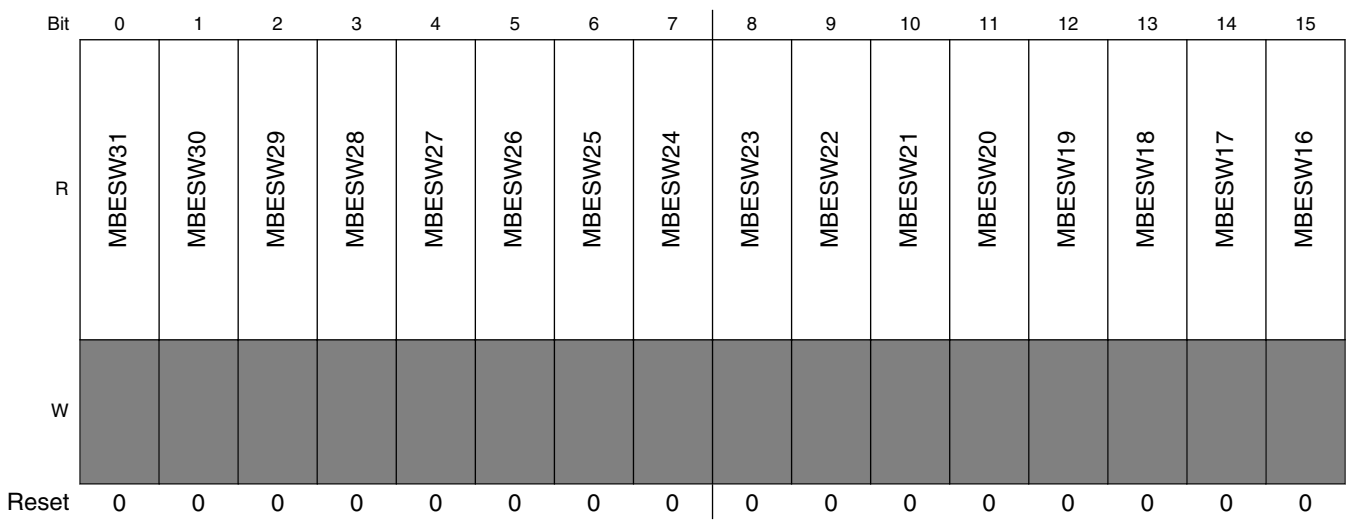
75.9.17 STCU2 On-Line MBIST End Flag Low Register (STCU2_MBELSW)

The STCU2_MBELSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 68h offset = 68h



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MBESW15	MBESW14	MBESW13	MBESW12	MBESW11	MBESW10	MBESW9	MBESW8	MBESW7	MBESW6	MBESW5	MBESW4	MBESW3	MBESW2	MBESW1	MBESW0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBELSW field descriptions

Field	Description
0 MBESW31	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
1 MBESW30	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
2 MBESW29	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
3 MBESW28	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
4 MBESW27	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
5 MBESW26	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
6 MBESW25	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
7 MBESW24	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBELSW field descriptions (continued)

Field	Description
8 MBESW23	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
9 MBESW22	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
10 MBESW21	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
11 MBESW20	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
12 MBESW19	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
13 MBESW18	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
14 MBESW17	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
15 MBESW16	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
16 MBESW15	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
17 MBESW14	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
18 MBESW13	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
19 MBESW12	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBELSW field descriptions (continued)

Field	Description
20 MBESW11	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBESW10	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBESW9	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBESW8	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBESW7	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBESW6	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBESW5	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBESW4	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBESW3	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBESW2	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBESW1	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBESW0	MBESWx: On-Line End status (0 to 31) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

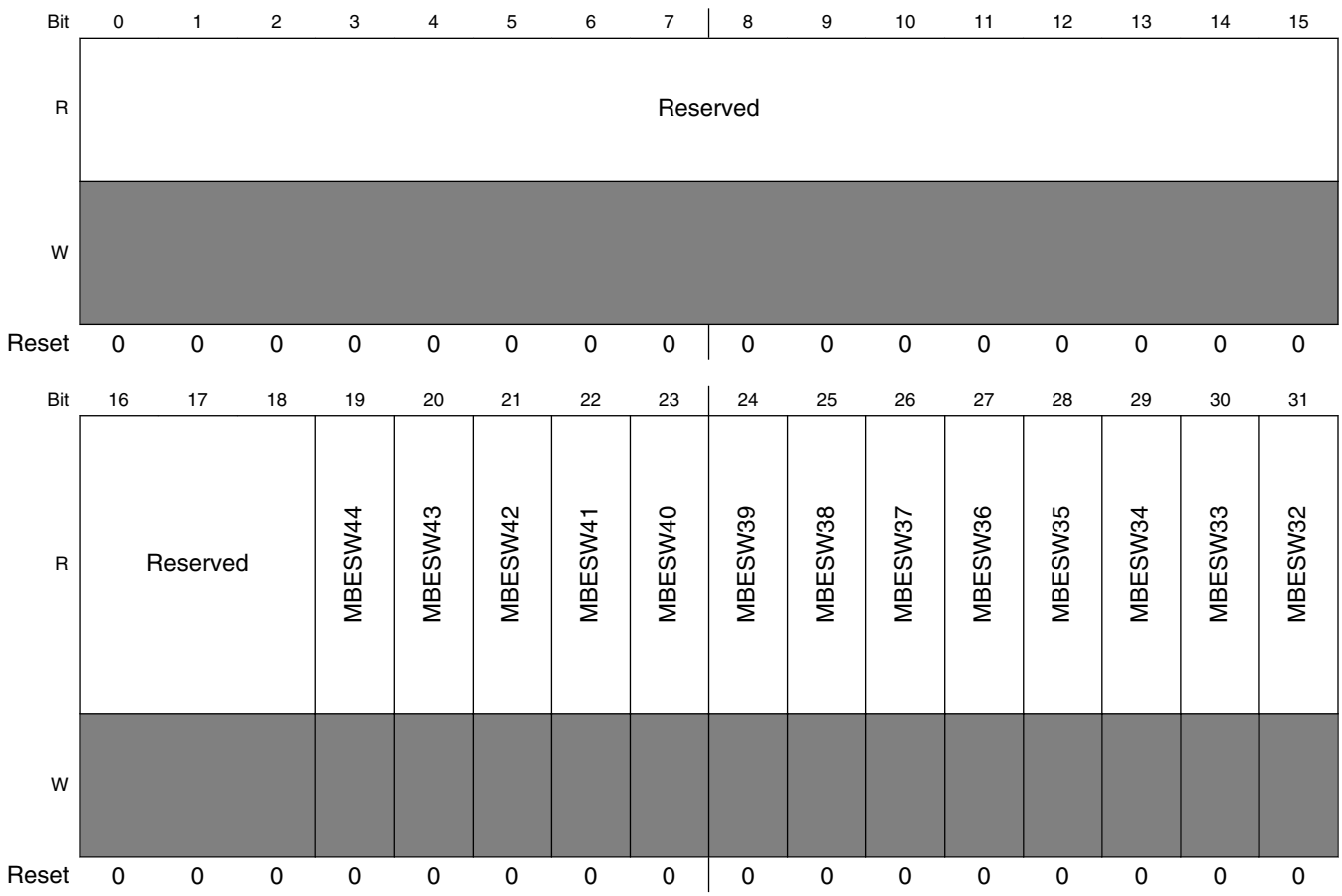
75.9.18 STCU2 On-Line MBIST End Flag Medium Register (STCU2_MBEMSW)

The STCU2_MBEMSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 32-63.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 6Ch offset = 6Ch



STCU2_MBEMSW field descriptions

Field	Description
0–18 Reserved	This field is reserved.
19 MBESW44	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

Table continues on the next page...

STCU2_MBEMSW field descriptions (continued)

Field	Description
20 MBESW43	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
21 MBESW42	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
22 MBESW41	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
23 MBESW40	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
24 MBESW39	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
25 MBESW38	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
26 MBESW37	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
27 MBESW36	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
28 MBESW35	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
29 MBESW34	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
30 MBESW33	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished
31 MBESW32	On-Line End status (32 to 63) of the selected MBIST 0 MBIST execution still ongoing 1 MBIST execution finished

75.9.19 STCU2 On-Line MBIST End Flag High Register (STCU2_MBEHSW)

The STCU2_MBEHSW register includes the End Flag related to the execution of the selected On-Line MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value as soon as STCU2_RUNSW[RUNSW] = 1.

Address: 0h base + 70h offset = 70h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

STCU2_MBEHSW field descriptions

Field	Description
0–31 Reserved	This field is reserved.

75.9.20 STCU2 MBIST Unrecoverable FM Low Register (STCU2_MBUFML)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFML register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 0-31.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Register description

Address: 0h base + 74h offset = 74h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MBUFM31	MBUFM30	MBUFM29	MBUFM28	MBUFM27	MBUFM26	MBUFM25	MBUFM24	MBUFM23	MBUFM22	MBUFM21	MBUFM20	MBUFM19	MBUFM18	MBUFM17	MBUFM16
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	MBUFM15	MBUFM14	MBUFM13	MBUFM12	MBUFM11	MBUFM10	MBUFM9	MBUFM8	MBUFM7	MBUFM6	MBUFM5	MBUFM4	MBUFM3	MBUFM2	MBUFM1	MBUFM0
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFML field descriptions

Field	Description
0 MBUFM31	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
1 MBUFM30	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
2 MBUFM29	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
3 MBUFM28	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
4 MBUFM27	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
5 MBUFM26	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
6 MBUFM25	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFML field descriptions (continued)

Field	Description
7 MBUFM24	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
8 MBUFM23	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
9 MBUFM22	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
10 MBUFM21	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
11 MBUFM20	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
12 MBUFM19	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
13 MBUFM18	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
14 MBUFM17	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
15 MBUFM16	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
16 MBUFM15	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
17 MBUFM14	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
18 MBUFM13	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFML field descriptions (continued)

Field	Description
19 MBUFM12	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
20 MBUFM11	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
21 MBUFM10	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
22 MBUFM9	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
23 MBUFM8	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
24 MBUFM7	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 MBUFM6	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 MBUFM5	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 MBUFM4	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 MBUFM3	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 MBUFM2	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 MBUFM1	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

Table continues on the next page...

STCU2_MBUFML field descriptions (continued)

Field	Description
31 MBUFM0	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

75.9.21 STCU2 MBIST Unrecoverable FM Medium Register (STCU2_MBUFMM)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFMM register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 32-63.

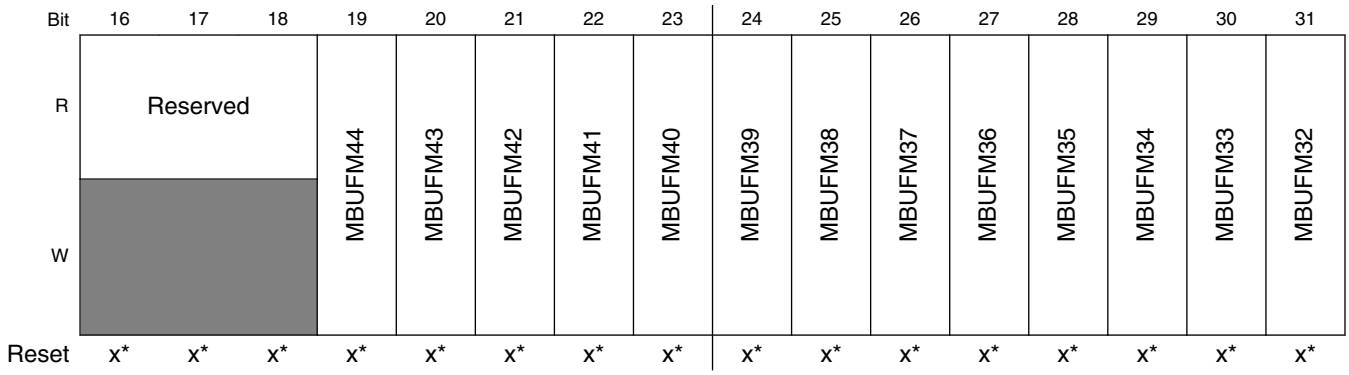
The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 78h offset = 78h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*



Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved			MBUFM44	MBUFM43	MBUFM42	MBUFM41	MBUFM40	MBUFM39	MBUFM38	MBUFM37	MBUFM36	MBUFM35	MBUFM34	MBUFM33	MBUFM32
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFMM field descriptions

Field	Description
0–18 Reserved	This field is reserved.
19 MBUFM44	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
20 MBUFM43	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
21 MBUFM42	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
22 MBUFM41	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
23 MBUFM40	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
24 MBUFM39	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
25 MBUFM38	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
26 MBUFM37	MBIST Unrecoverable Fault Mapping

Table continues on the next page...

STCU2_MBUFMM field descriptions (continued)

Field	Description
	0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
27 MBUFM36	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
28 MBUFM35	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
29 MBUFM34	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
30 MBUFM33	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping
31 MBUFM32	MBIST Unrecoverable Fault Mapping 0 Recoverable Fault mapping 1 Unrecoverable Fault mapping

75.9.22 STCU2 MBIST Unrecoverable FM High Register (STCU2_MBUFMH)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MBUFMH register defines the fault mapping, in terms of Unrecoverable or Recoverable fault, of the MBIST in the range NMCUT = 64-95.

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

Address: 0h base + 7Ch offset = 7Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MBUFMH field descriptions

Field	Description
0–31 Reserved	This field is reserved.

75.9.23 STCU2 MBIST Control Register (STCU2_MB_CTRL*n*)

NOTE

The reset value is chip-specific; see the chapter that describes how modules are configured and connected.

The STCU2_MB_CTRL register defines the control setting of MBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when either of the following conditions is true:

- Off-line Self-Test phase is active
- On-line Self-Test phase is active and STCU2_CFG[WRP] = 0

The offset of this register is a function of NMCUT .

Address: 0h base + 600h offset + (4d × i), where i=0d to 44d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CSM	PTR							0							
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0															
W																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value is chip-specific; see the chapter that describes how modules are configured and connected.x = Undefined at reset.

STCU2_MB_CTRL*n* field descriptions

Field	Description
0 CSM	Concurrent/sequential mode 0 Sequential mode 1 Concurrent mode
1–7 PTR	Next MBIST pointer PTR defines the logical pointer to the next MBIST to be scheduled. The next MBIST is scheduled concurrently to the current one if the CSM bit is set to 1; otherwise it is scheduled sequentially to the completion of the current MBIST execution. In case of NIL pointer the CSM bit must be set Sequential (0) to define this is the end of the list. The self-testing procedure is stopped when the current MBIST has been completed.
8–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

75.10 Use cases and limitations

The STCU2 module is designed to give high flexibility during the on/off-line test. In the following are reported the programming operations that have to be performed to correctly use the functionality of this module considering the three typical usage cases:

- the Off-line Self Test Sequence after a reset trigger event
- the bypass mode of the Self Test Sequence after a reset trigger event
- the On-line Self-test Sequence

75.10.1 Offline self-test sequence

This is the typical mode of using the STCU2 module after reset trigger event is applied to STCU2. The SSCM DCF bus is used to retrieve the STCU2 schedule and MBIST execution parameters stored into the NVM memory.The target is to cover the amount of physical defects in the digital logic and in the System RAMs/ROMs according to the System specifications. The sequence is the following:

- Unlock the Off-Line STCU2 access writing the Key1/Key2 sequence into the STCU2_SKC register
- Program the MBIST fault reaction conditions (UF/RF) setting, depending on the number of memories, the STCU2_MBUFML/M/H registers
- Program the STCU2 internal fault reaction conditions (UF/RF) setting the STCU2_ERR_FM register
- Program the STCU2_MB_CTRL registers of each NMCUT to be executed
- Program the STCU2_WDG register to define the time budget assigned for the MBIST execution

- Program the STCU2_CFG register to define: the core and MBIST TCK clock prescaling factor setting the CLK_CFG bits, the PMOSEN or MBU in order to define the algorithm to be used during MBIST run, and finally set the pointer to the first NMCUT to be executed
- In case the PLL usage has been enabled, program the STCU2_PLL_CFG value in order to set the required system frequency
- Program the RUN bit into the STCU2_RUN register to enable the Off-Line Self-Test execution and the MBPLLEN bits to enable the PLL during MBIST execution according to STCU2_PLL_CFG register content
- Start the Off-Line Self-Test and, depending on PLL management and MBIST execution order, when PLL LOCK is high.
- Wait the WDGEOC run time execution specified into the STCU2_WDG register.
- The STCU2 switches off the core clock at the end of the Self-Test run and releases the PLL control in case this feature has been enabled.
- The STCU2 resets the STCU2_RUN[RUN] bit and the device exits from boot sequence. In case there are failures, the STCU2 flags these failures toward the FCCU.
- The software application reads the STCU2_MBSL/M/H flag registers, depending on the number of memories, to check the failed RAMs/ROMs Memory BIST when UF or RF is/are detected.
- The software application reads the STCU2_MBEL/M/H flag registers, depending on the number of memories, to check if the RAMs/ROMs Memory BIST is still on-going when UF or RF is/are detected.
- The software application reads the bits INV, ENGE, WDTO, LOCKE of the STCU2_ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF is/are detected.

NOTE

The software application reads operations on the STCU2 register after off-line self test execution has been successfully performed are just an additional reliability layer added to improve the already implemented Auto Self-Test feature included into STCU2.

75.10.2 Online self test sequence

The online self test sequence is the most commonly used STCU2 application mode. The IPS interface is used to program the STCU2 registers and to schedule MBIST execution. Since online MBIST operations clean-up the content of RAM (MBIST) tested, activation of this mode must be approached with caution. The suggested sequence is the following:

1. Ensure the RUN_CFG field value in all implemented MC_ME_PCTL*n* registers forces the associated peripheral to a frozen, clock gated state.
2. Write the Key1/Key2 sequence to the SKC register to unlock online STCU2 access.
3. Ensure CFG[WRP]=0 to enable IPS bus access to the online self test registers.
4. If required, program the following registers to select the MBIST and STCU2 internal fault reaction: MBUFML/M/H and ERR_FM.

NOTE

The MBIST and STCU2 internal fault reaction should have been already programmed during the offline self test sequence and so there should be no need to program again. However, in case they have not been programmed or there is the need of changing them for any reason, they can be overwritten.

5. Overwrite/Program the MB_CTRL registers of each NMCUT to be executed.
6. Overwrite/Program the WDG register to define the time budget assigned for the MBIST execution
7. Overwrite/Program the CFG register to define: the core and MBIST TCK clock prescaling factor setting the CLK_CFG bits, the PMOSEN or MBU in order to define the algorithm to be used during MBIST run, and finally set the pointer to the first NMCUT to be executed.
8. Program the RUNSW[RUNSW] field to enable online self test execution; the MBSWPLLEN bits to enable the PLL lock signal monitor during MBIST execution
9. Start the online self test and depending on PLL management and MBIST execution order, also when PLL LOCK is high.
10. Wait the WDGEOC run time execution specified into the WDG register
11. The STCU2 switch-off the core clock at the end of the self test run
12. The STCU2 resets the RUNSW[RUNSW] field. In both the reset cases and in case of fault/s, the STCU2 flags the failure toward the FCCU.
13. The software application reads the MBSLSW/MSW/HSW flag registers, depending on the number of memories, to check the failed RAMs/ROMs Memory BIST when UF or RF is/are detected.
14. The software application reads the MBELSW/MSW/HSW flag registers, depending on the number of memories, to check if the RAMs/ROMs Memory BIST is still on-going when UF or RF is/are detected.

15. The software application reads the bit INVPSW, ENGESW, WDTOSW, LOCKESW of the ERR_STAT register to check whether there has been an internal STCU2 engine/parameters failure when UF or RF is/are detected.

NOTE

1. A reset is necessary after running any kind of online self test.
2. The software application read operations on the STCU2 register after successful online self test execution are an additional reliability layer to improve the auto-self-test feature included in the STCU2.
3. Flash memory must be in the idle state when BIST execution is started. Program and erase functionality must not be enabled when entering BIST execution.

75.10.3 Bypass USER Mode

In this case, the USER application parameters stored in FLASH and read by SSCM are written in order to skip the Self-Test procedure after a reset trigger event is applied. It might be useful in case the device is not configured for applications requiring improved reliability. The sequence is the following:

1. Unlock the STCU2 access writing the Off-Line Key1/Key2 sequence into the STCU2_SKC register
2. Set the BYP bit in the STCU2_RUN register

After BYP bit is set, the core clock is switched off, the Self-Test sequence is not applied and the system can wake up and start the user application.

75.10.4 Design implementation information

The following list reports the limitations related to the current implementation:

- Every NMCUT BIST can be run once during the online or offline self-test procedure.
- Because software has full control of the system while applications are running during the online self-test, STCU2 is not allowed to take control of the PLL but can just monitor the lock signal to flag wrong unlock events.

Chapter 76
Cyclic Redundancy Check (CRC) Unit

76.1 Introduction

The Cyclic Redundancy Check (CRC) computing unit is dedicated to the computation of CRC, off-loading the CPU. Each context has a separate CRC computation engine in order to allow the concurrent computation of the CRC of multiple data streams. The CRC computation is performed at speed without wait-state insertion. Bit-swap and bit-inversion operations can be applied on the final CRC signature. Each context can be configured with one of three hard-wired polynomials, normally used for most of the standard communication protocols. The data stream supports multiple data width (byte/halfword/word) formats.

76.2 Main features

- Single context is supported
- Zero-wait states during the CRC computation (pipeline scheme)
- Four hard-wired polynomials (two CRC-8, CRC-32 ITU-T V.42, and CRC-16-CCITT)
- Support for byte, halfword, or word width of the input data stream

76.2.1 Standard features

- Peripheral bus interface
- CRC-8 VDA CAN
- CRC-16-CCITT

- CRC-32 ITU-T V.42 (see note below)
- CRC-8-H2F Autosar polynomial

NOTE

Although the CRC-32 ITU-T V.42 polynomial is used for CRC generation, the generation algorithm differs (MISR) from the one used by the IEEE 802.3 standard.

76.3 Block diagram

The top level diagram of the CRC module is given in the following figure. Refer to Main features for number of contexts in this module.

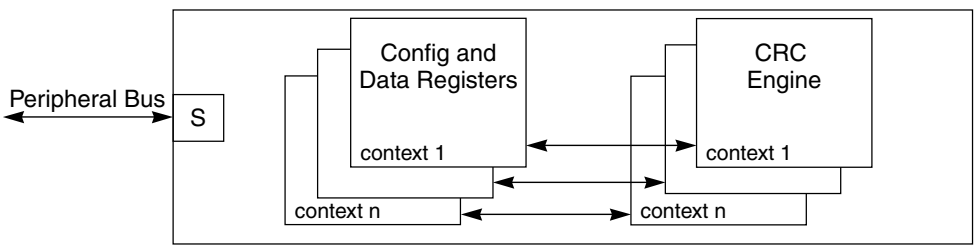


Figure 76-1. CRC top level diagram

76.4 External signal description

The CRC module does not generate any external signals.

76.4.1 Peripheral bus interface

The peripheral bus interface is a slave bus used for configuration and data streaming (CRC computation) purposes via CPU or DMA. The following bus operations (contiguous byte enables) are supported:

- Word (32 bits) data write/read operations to any registers
- Low and high halfword (16 bits, data[31:16] or data[15:0]) data write/read operations to any registers

- Byte (8 bits, data[31:24], data[23:16], data[15:8], or data[7:0]) data write/read operations to any registers
- Any other operation (free byte enables or other operations) must be avoided.

The CRC module generates a transfer error in the following cases:

- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register

The registers of the CRC module are read/write accessible in each access mode:
- user
- supervisor

The following summarizes bus operation performance:

- Zero wait states (single bus cycle) for each write/read operation to the CRC_CFG and CRC_INP registers
- Zero wait states (single bus cycle) for each write operation to the CRC_ CSTAT register
- Double wait states (3 bus cycles) for each read operation to the CRC_ CSTAT or CRC_OUTP registers immediately following (next clock cycle) a write operation to the CRC_CSTAT, CRC_INP, or CRC_CFG registers belonging to the same context; in all the other cases, no wait states are inserted

76.5 CRC memory map and registers

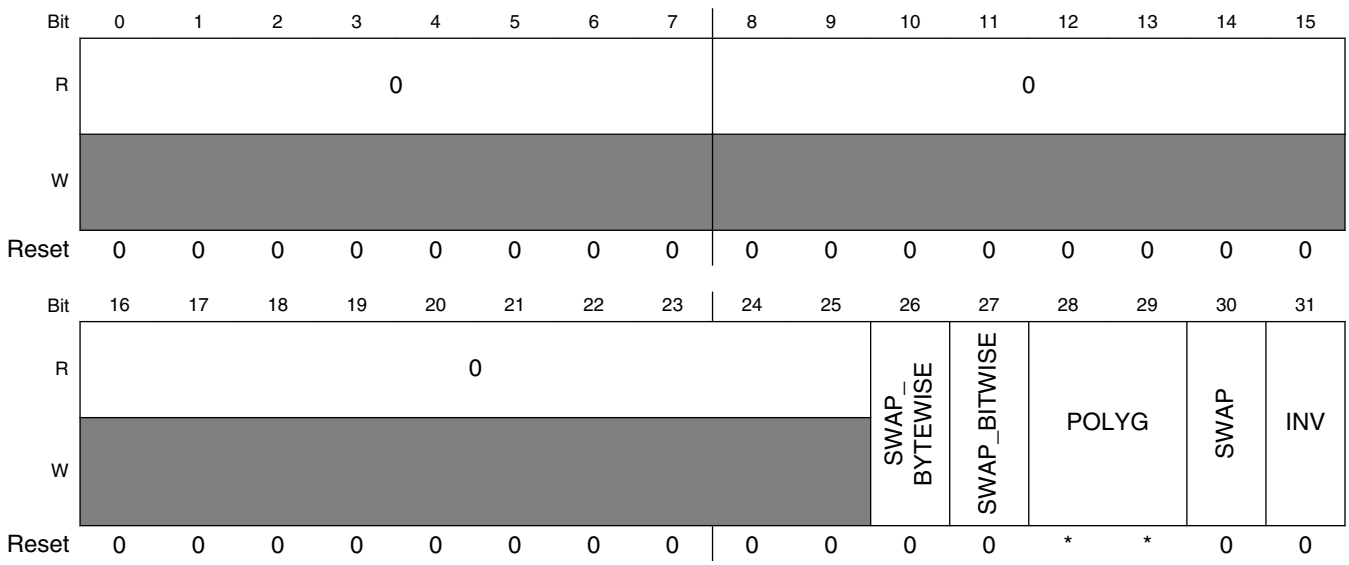
CRC memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Configuration Register (CRC_CFG)	32	R/W	See section	76.5.1/3276
4	Input Register (CRC_INP)	32	R/W	0000_0000h	76.5.2/3277
8	Current Status Register (CRC_CSTAT)	32	R/W	FFFF_FFFFh	76.5.3/3278
C	Output Register (CRC_OUTP)	32	R	FFFF_FFFFh	76.5.4/3278

76.5.1 Configuration Register (CRC_CFG)

Access: User read/write.

Address: 0h base + 0h offset = 0h



* Notes:

- POLYG field: Reset value is a function of n in CRC_CFGn. When n%2 = 0, reset value = 01b. When n%2 = 1,reset value = 00b. So CRC_CFG1 POLYG reset value = 00b (Since 1%2=1).

CRC_CFG field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 SWAP_ BYTEWISE	Swap CRC_INP byte-wise **NOTE:** INV and SWAP bits are set to 1 for CRC-32 polynomial calculations. 0 Do not swap 1 Perform byte-wise swap on CRC_INP input data internally for CRC-32 polynomial calculations.
27 SWAP_BITWISE	Swap CRC_INP bit-wise 0 Do not swap 1 Perform bit-wise swap on CRC_INP input data internally for CRC-8 and CRC-16 polynomial calculations.
28–29 POLYG	Polynomial selection

Table continues on the next page...

CRC_CFG field descriptions (continued)

Field	Description
	This bit can be written only during the configuration phase. 00 CRC-CCITT polynomial 01 CRC-32 polynomial 10 CRC-8 polynomial 11 CRC-8-H2F Autosar polynomial
30 SWAP	Swap selection This bit can be written only during the configuration phase. The swap operation is a bit-by-bit swapping of the content. 0 No swap selection applied on the CRC_OUTP content 1 Swap selection (MSB to LSB, LSB to MSB) applied on the CRC_OUTP content; in case of CRC-CCITT polynomial, the swap operation is applied on the 16 least-significant bits
31 INV	Inversion selection This bit can be written only during the configuration phase. The inversion operation is a complement (or negation) of the content. 0 No inversion selection applied on the CRC_OUTP content 1 Inversion selection (bit x bit) applied on the CRC_OUTP content

76.5.2 Input Register (CRC_INP)

Access: User read/write.

NOTE

High level code must be written in a way to realize byte- , halfword- and word-wide writes on assembler code level. Any endianness problem faced can be corrected by appropriate configuration of CRC_CFG[SWAP_BYTEWISE] and CRC_CFG[SWAP_BITWISE].

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	INP																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CRC_INP field descriptions

Field	Description
0–31 INP	Input data for the CRC computation This register can be written with word-, halfword- (high or low), or byte-wide writes in any sequence. Only the bits written are fed to the CRC engine. In case of halfword write operation, the bytes must be contiguous.

76.5.3 Current Status Register (CRC_CSTAT)

Access: User read/write.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CSTAT																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CRC_CSTAT field descriptions

Field	Description
0–31 CSTAT	CRC signature status This register includes the current status of the CRC signature. No bit swap and inversion are applied to this register. In the case of the CRC-CCITT polynomial, only the 16 least-significant bits are significant. The 16 most-significant bits are set to 0 during the computation. In the case of the CRC-8 polynomial, only the 8 least-significant bits are significant. The 24 most-significant bits are set to 0 during the computation. The CSTAT register can be written at byte, halfword, or word. This register can be written only during the configuration phase.

76.5.4 Output Register (CRC_OUTP)

Access: User read-only.

Address: 0h base + Ch offset = Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	OUTP																															
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CRC_OUTP field descriptions

Field	Description
0–31 OUTP	Final CRC signature This register includes the final signature corresponding to the CRC_CSTAT register value eventually swapped and inverted. In the case of the CRC-CCITT polynomial, only the 16 least-significant bits are significant. The 16 MSB bits are set to 0 during the computation. In the case of the CRC-8 polynomial, only the 8 least-significant bits are significant. The 24 most-significant bits are set to 0 during the computation.

76.6 Functional description

The CRC module supports the CRC computation for each context. Each context has its own complete set of registers, including the CRC engine. The data flow of each context can be interleaved. The data stream can be structured as a sequence of bytes, halfwords, or words. The input data sequence is provided, eventually mixing the data formats (byte, halfword, and word), writing to the input data register (CRC_INP).

The data stream is generally executed by *n* concurrent DMA data transfers (mem2mem) where *n* is less or equal to the number of contexts.

The standard generator polynomials are given in Equation 46 on page 3279, Equation 47 on page 3279, and Equation 48 on page 3279 for the CRC computation of each context. Two separate registers are available: CRC_INP for writing data and CRC_CSTAT for retrieving the (accumulated up to now) CRC.

$$x^8 + x^4 + x^3 + x^2 + 1$$

Equation 46. CRC8 VDA CAN

$$x^{16} + x^{12} + x^5 + 1$$

Equation 47. CRC-CCITT (x.25 protocol)

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Equation 48. CRC-32 (ITU-T V.42 protocol)

$$x^5 + x^3 + x^2 + x + 1$$

Equation 49. CRC-8-H2F (Autosar 4.0 protocol)

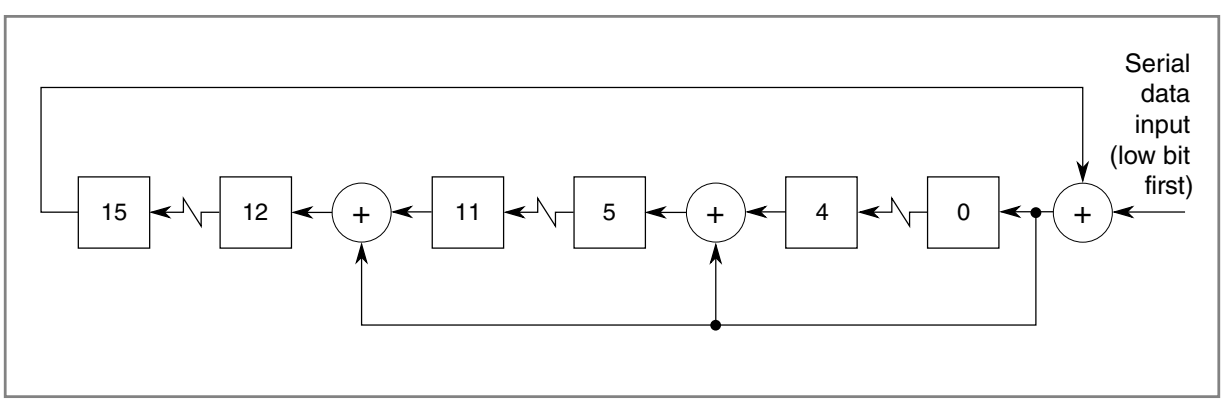


Figure 76-2. CRC-CCITT engine concept scheme

The initial seed value of the CRC can be programmed by initializing the CRC_CSTAT register. A diagram illustrating serial data loading of the CRC engine is shown in Figure 76-2 for the CRC-CCITT. Note that, conceptually, the least-significant, or "right-most," bit shown in Input Register (CRC_INP), is fed first into the engine. The actual implementation executes the CRC computation in a single clock cycle (parallel data loading). A pipeline scheme has been adopted to decouple the IPS bus interface from the CRC engine in order to allow the computation of the CRC at speed (zero wait states).

If the CRC signature is used for encapsulation in the data frame of a communication protocol (for example, SPI, etc.), a bit swap (high bit for low bit or low bit for high bit) and/or bit inversion of the final CRC signature can be applied to CRC_OUTP before transmitting the CRC.

Use of the CRC module is summarized in the flow chart given in Figure 76-3.

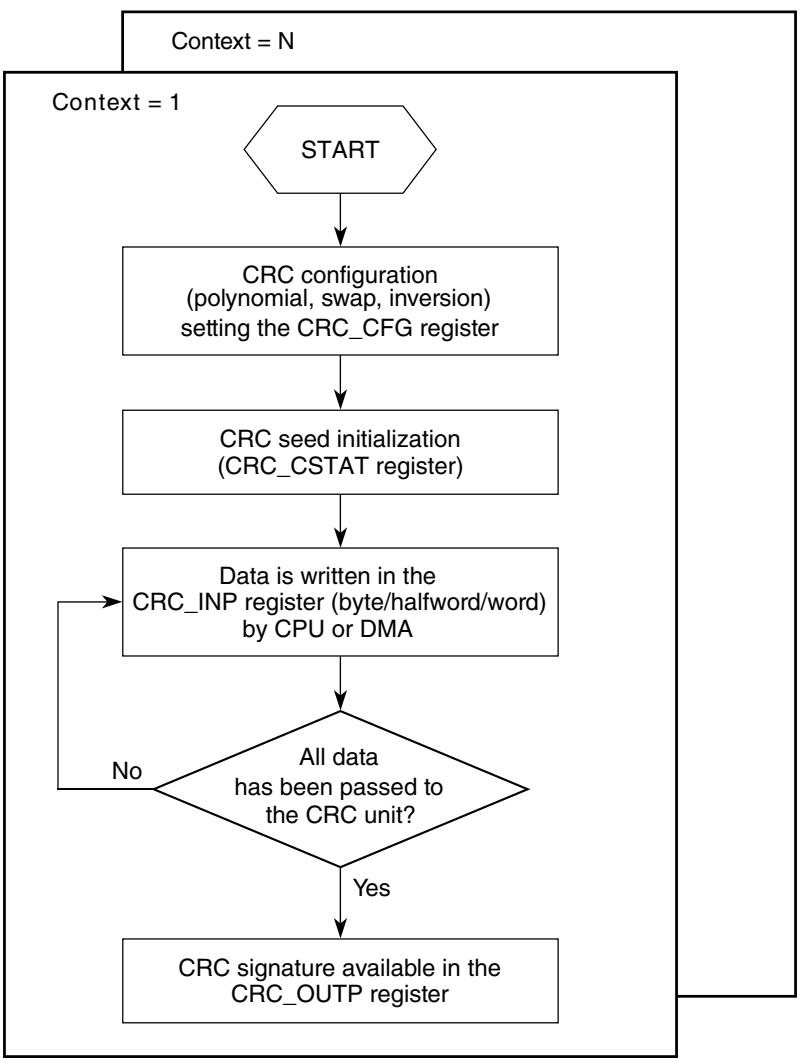


Figure 76-3. CRC computation flow

76.7 Use cases

76.7.1 Programming example

The number of contexts depends on the application. The overall number of contexts for the CRC peripheral depends on the number of peripherals that concurrently require intervention by the CRC module. Two main use cases are considered:

- Calculation of the CRC of the configuration registers during the process safety time
- Calculation of the CRC on the incoming/outcoming frames for the communication protocols (not protected with CRC by definition of the protocol itself) used as a safety-relevant peripheral

The signature of the configuration registers is computed correctly only if these registers do not contain any status bit.

Assuming that the DMA engine has *n* channels (greater than or equal to the number of contexts) configurable for the following types of data transfer—mem2mem, periph2mem, mem2periph—the following sequence, as shown in the following figure, is applied to manage the transmission data flow:

1. DMA/CRC module configuration (context x, channel x) by CPU
2. Payload transfer from the MEM to the CRC module (CRC_INP register) after MSB-to-LSB change (input mirroring) to calculate the CRC signature (phase1) by DMA (mem2mem data transfer, channel x)
3. CRC signature copy from the CRC module (CRC_OUTP register) to the MEM (phase 2) by CPU
4. Data block (payload + CRC) transfer from the MEM to the PERIPH module (for example, SPI Tx FIFO) (phase 3) by DMA (mem2periph data transfer, channel x)

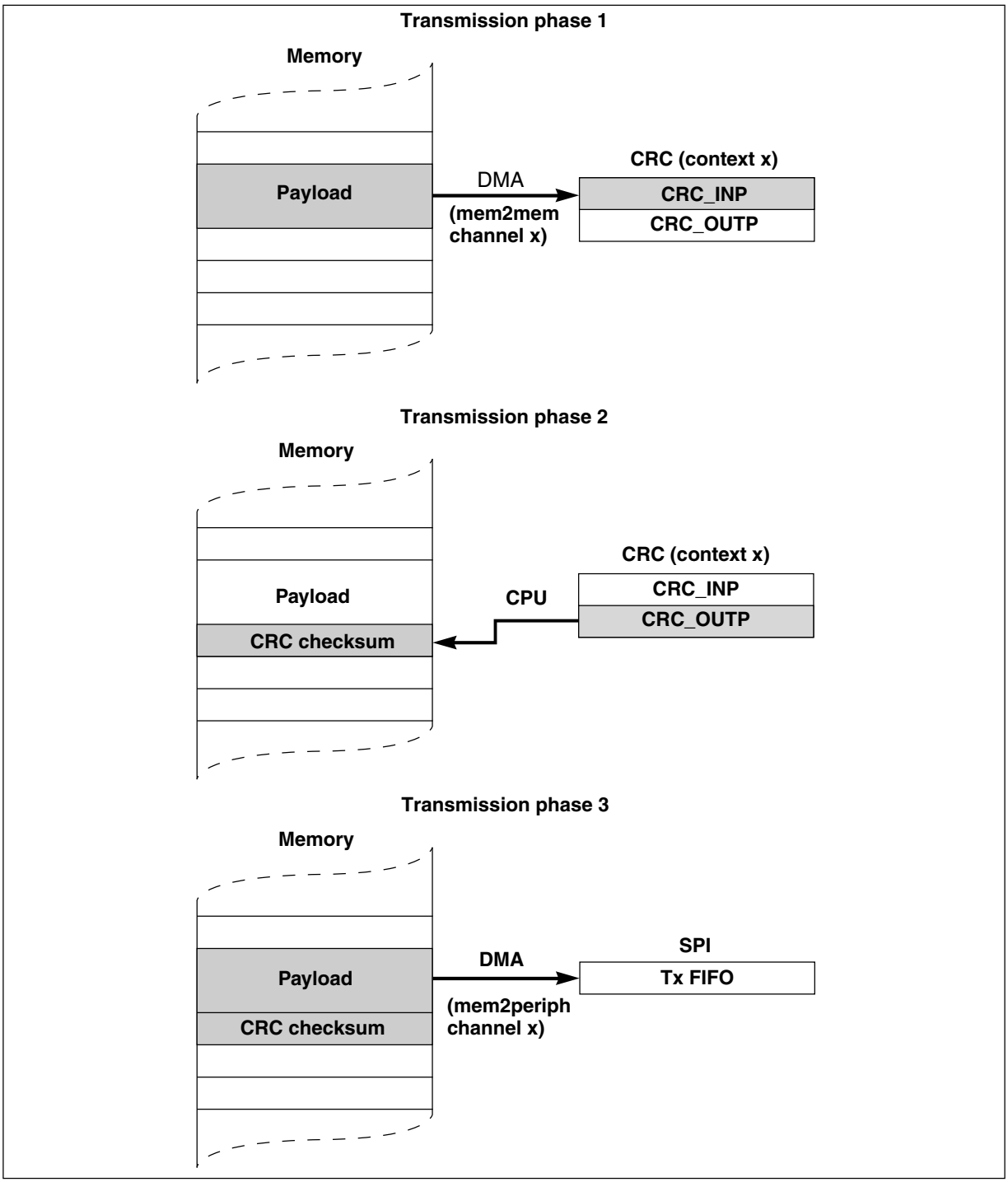


Figure 76-4. DMA-CRC transmission sequence

76.7.2 Register programming

- INV and/or SWAP (affecting the output only) can be applied to CRC-8, CRC-16, and CRC-32. Both can be applied together.

- SWAP_BITWISE (affecting the input only) can be applied for all polynomials if required by an application.
- SWAP_BYTEWISE (affecting the input only) can be applied for CRC-16 and CRC-32 polynomials only.
- SWAP_BITWISE has priority over SWAP_BYTEWISE when both are applied together.
- When generating CRC-32 for the ITU-T V.42 standard the user needs to set SWAP_BYTEWISE together with INV and SWAP.
- When generating CRC-16 the user needs to set SWAP_BITWISE bit.

Chapter 77
Register Protection

77.1 Chip-specific Register Protection information

77.1.1 Register protection configuration

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

Table 77-1. Related information

Topic	Related module
System memory map	All
SIUL registers	SIUL
PMC Digital Interface (PMCDIG) registers	PMCDIG
Mode Entry (MC_ME) module registers	MC_ME
PLL Digital Interface (PLLDIG) registers	PLLDIG
Clock Monitor Unit (CMU) registers	CMU
Fast Oscillator Digital Interface (FXOSC) registers	FXOSC
Slow External Oscillator (SXOSC) registers	SXOSC
Memory Error Management Unit (MEMU_0 and MEMU_1) registers	MEMU
Clock Generation Module (MC_CGM) registers	MC_CGM
Reset Generation Module (MC_RGM) registers	MC_RGM
LPU controller (LPU_CTL) registers	LPU_CTL

NOTE

This following restriction applies to the CGM memory mapped registers (FIRC, FXOSC , SIRC, SXOSC and CMU) which are usable in LPU mode:

1. Access to these registers should be unlocked via Register Protection configuration in RUN/DRUN modes before LPU modes entry to allow access to modify these registers OR.
2. If Access to these registers are locked via Register Protection configuration in RUN/DRUN modes before LPU modes entry, then don't access these registers in LPU modes as it would generate exception.

77.1.2 Clock Monitor Unit (CMU) protected registers

The following table lists the Clock Monitor Unit (CMU) module registers that can be protected.

Table 77-2. Protected CMU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CMU_CSR	32	0h	8 (Byte 3)
CMU_HFREFR	32	8h	16 (Bytes 2 and 3)
CMU_LFREFR	32	Ch	16 (Bytes 2 and 3)
CMU_MDR	32	18h	32

77.1.3 Fast Oscillator Digital Interface (FXOSC) protected registers

The following table lists the Fast Oscillator Digital Interface (FXOSC) module registers that can be protected.

Table 77-3. Protected FXOSC registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
FXOSC_CTL	32	0h	32

77.1.4 Clock Generation Module (MC_CGM) protected registers

The following table lists the Clock Generation Module (MC_CGM) module registers that can be protected.

Table 77-4. Protected MC_CGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
CGM_SC_DC0	32	0x07E8h[1]	32
CGM_SC_DC1	32	0x07ECh[1]	32
CGM_SC_DC2	32	0x07F0h[1]	32
CGM_SC_DC5	32	0x7FCh[1]	32
CGM_CLOCKOUT1_SC	32	0x0180	32
CGM_AC2_SC	32	0x0840	Byte 0
CGM_AC3_SC	32	0x0860	Byte 0
CGM_AC4_SC	32	0x0880	Byte 0
CGM_AC5_SC	32	0x01C0	32
CGM_AC6_SC	32	0x08C0	Byte 0
CGM_AC8_SC	32	0x0280	32
CGM_AC9_SC	32	0x02C0	32

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

77.1.5 Mode Entry (MC_ME) protected registers

The following table lists the Mode Entry (MC_ME) module registers that can be protected.

Table 77-5. Protected MC_ME registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ME_MCTL	32	4h	32
ME_ME	32	8h	32
ME_IM	32	10h	32
ME_SAFE_MC	32	28h	32
ME_DRUN_MC	32	2Ch	32
ME_RUN0_MC	32	30h	32
ME_RUN1_MC	32	34h	32
ME_RUN2_MC	32	38h	32
ME_RUN3_MC	32	3Ch	32

Table continues on the next page...

Table 77-5. Protected MC_ME registers (continued)

Register	Register size (bits)	Offset from module base address	Protected size (bits)
ME_RUN_PC0-7	32	80h-9Ch	32
ME_PCTL0-2	8	C0-C2h	8
ME_PCTL10-12[1]	8	CA-CCh	8
ME_PCTL15[1]	8	CFh	8
ME_PCTL20-22[1]	8	D4-D6h	8
ME_PCTL24-25[1]	8	D8-D9h	8
ME_PCTL28[1]	8	DCh	8
ME_PCTL30-33[1]	8	DE-E1h	8
ME_PCTL40-43[1]	8	E8-EBh	8
ME_PCTL50-65[1]	8	F2-101h	8
ME_PCTL70-77[1]	8	106-10Dh	8
ME_PCTL82-84[1]	8	112-114h	8
ME_PCTL90-91[1]	8	11A-11Bh	8
ME_PCTL93[1]	8	11Dh	8
ME_PCTL96-99[1]	8	120-123h	8
ME_PCTL102[1]	8	126h	8
ME_PCTL105[1]	8	129h	8
ME_CCTL1	16	1C6h	16
ME_CCTL3[2]	16	1CAh	16

1. See the module memory map in the MC_ME chapter for implemented ME_PCTL
2. See the module memory map in the MC_ME chapter for implemented ME_CCTL*n*.

77.1.6 Reset Generation Module (MC_RGM) protected registers

The following table lists the Reset Generation Module (MC_RGM) module registers that can be protected.

Table 77-6. Protected MC_RGM registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
RGM_DBRE	32	30h	32
RGM_FERD	32	310h	32
RGM_FEAR	32	320h	32
RGM_FBRE	32	330h	32
RGM_FESS	32	340h	32

77.1.7 Memory Error Management Unit (MEMU) protected registers

The following table lists the Memory Error Management Unit (MEMU) module registers that can be protected.

Table 77-7. Protected MEMU registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
MEMU_0			
MEMU_0_CTRL	8	0h	8 (Byte 2)
MEMU_0_DEBUG	32	Ch	32
MEMU_1			
MEMU_1_CTRL	8	0h	8 (Byte 2)
MEMU_1_DEBUG	32	Ch	32

77.1.8 PLL Digital Interface (PLL_DIG) protected registers

The following table lists the PLL Digital Interface (PLLDIG) module registers that can be protected.

Table 77-8. Protected PLLDIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PLLDIG_PLLCR	32	20h	16 (Bytes 2 and 3)
PLLDIG_PLLDV	32	28h	32

77.1.9 Power Management Controller digital interface (PMCDIG) protected registers

The following table lists the PMCDIG module registers that can be protected.

Table 77-9. Protected PMCDIG registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
PMCDIG_RDCR	32	0h	32
PMCDIG_MCR	32	10h	32

77.1.10 SIUL2 protected registers

The following table lists the SIUL2 registers that can be protected.

Table 77-10. Protected SIUL2 registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
SIUL2_IRER0	32	18h	32
SIUL2_IRSR0	32	20h	32
SIUL2_IREER0	32	28h	32
SIUL2_IFEER0	32	30h	32
SIUL2_IFER0	32	38h	32
SIUL2_IFMCR[0-31][1]	32	40h-BCh	32
SIUL2_IFCPR	32	C0h	32
SIUL2_MSCR[0-511] SIUL2_IMCR[0-511] [2]	32	240h-65Ch[3] A40h-123Ch	n x 32
SIUL2_GPDO[0-511][4]	8	1300h-1404h[3]	n x 8
SIUL2_PGPDO[0-31][5]	16	1700h-173Eh[3]	n x 16
SIUL2_MPGPDO[0-31][6]	32	1780h-17FCh[3]	n x 32

1. All available IFMCR*n* registers are protected
2. All available MSCR*n* and IMCR*n* registers are protected except IMCR 506, IMCR 507, IMCR 508 and IMCR 509
3. See the module memory map in the SIUL2 chapter for address offset
4. All available GPDO*n* registers are protected
5. All available PGPDO*n* registers are protected
6. All available MPGPDO*n* registers are protected

77.1.11 Slow Oscillator Digital Interface (SXOSC) protected registers

The following table lists the Slow Oscillator Digital Interface (SXOSC) module registers that can be protected.

Table 77-11. Protected SXOSC registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
SXOSC_CTL	32	0h	32

77.1.12 LPU Controller (LPU_CTL) protected registers

The following table lists the LPU Controller (LPU_CTL) module registers that can be protected.

Table 77-12. Protected LPU_CTL registers

Register	Register size (bits)	Offset from module base address	Protected size (bits)
LPU_MCTL	32	0h	32
LPU_RUN_CF	32	4h	32
LPU_STOP_CF	32	8h	32
LPU_STBY_CF	32	0Ch	32
LPU_ICR	32	14h	32
LPU_MDIS	32	18h	32

77.2 Overview

The Register Protection (REG_PROT) module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The address locations that can be protected are module-specific.

The protection module is located between the module under protection and the peripheral bridge (PBRIDGE/AIPS-Lite). This is shown in the following figure.

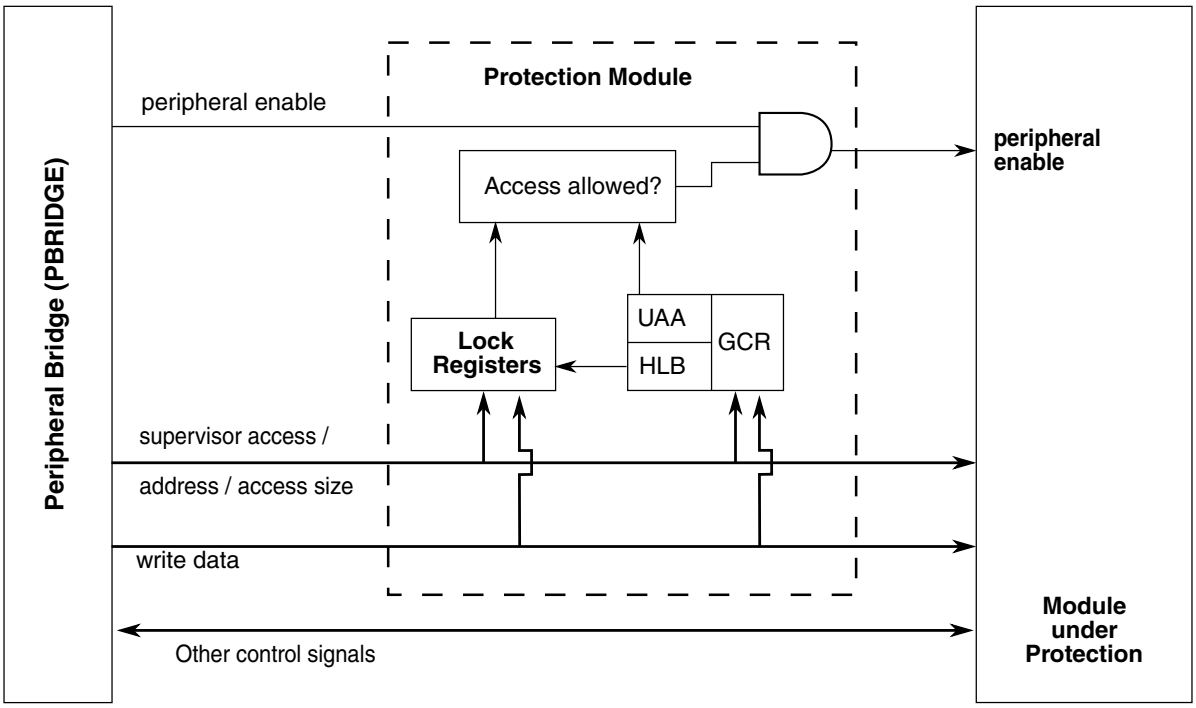


Figure 77-1. REG_PROT Block Diagram

77.3 Features

Register Protection includes these distinctive features:

- Restrict write accesses for the module under protection to supervisor mode only
- Lock registers for first 6 KB of memory-mapped address space
- Write to address mirror automatically sets corresponding lock bit
- Once configured lock bits can be protected from changes

77.4 Modes of operation

The Register Protection module is operable when the module under protection is operable.

77.5 External signal description

There are no external signals.

77.6 Memory map and register definition

This section provides a detailed description of the memory map of a module with register protection. The original 16 KB module memory space is divided into five areas as shown in the following figure.

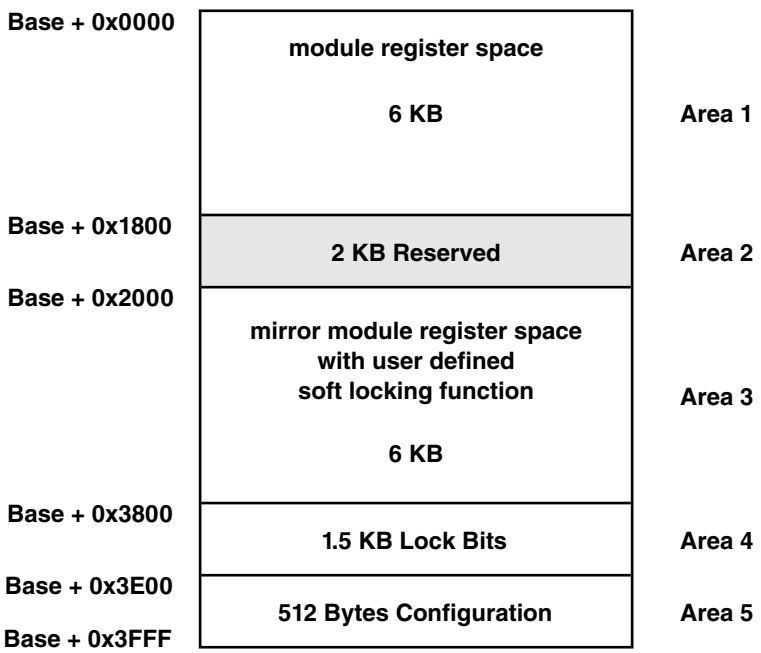


Figure 77-2. REG_PROT Memory Diagram

- Area 1 is 6 KB and holds the normal functional module registers and is transparent for all read/write operations.
- Area 2 is 2 KB starting at offset 0x1800 is a reserved area, which shall not be accessed.
- Area 3 is 6 KB, starting at offset 0x2000 and is a mirror of area 1. A read/write access to an offset 0x2000+X will read/write the register at offset X. However a write access to offset 0x2000+X will additionally set the optional Soft Lock Bits for this offset X in the same cycle as the register at offset X is written. This provides for an automatic write and lock operation. Not all registers in area 1 need to have protection defined by associated Soft Lock Bits. For unprotected registers at offset Y, accesses to offset 0x2000+Y will be identical to accesses at offset Y.

- Area 4 is 1.5 KB and holds the Soft Lock Bits, one bit per byte in area 1. The four Soft Lock Bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.
- Area 5 is 512 bytes large and holds the configuration bits of the protection mode. There is one configuration hard lock bit per module that prevents all further modifications to the Soft Lock Bits and can only be cleared by a system reset once set. The other bits, if set, will allow user access to the protected module.

If any locked byte is accessed with a write transaction, a transfer error will be issued to the system and the write transaction will not be executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32-bit registers in Areas 4 and 5 will result in a transfer error.

77.6.1 Memory map

Following is the memory map for a module which is protected via a REG_PROT module.

Table 77-13. Generic Protected Module Memory Map

Offset	Use
0x0000	Module Register 0 (MR0)
0x0001	Module Register 1 (MR1)
0x0002	Module Register 2 (MR2)
0x0003 - 0x17FF	Module Register 3 (MR3) - Module Register 6143(MR6143)
0x1800 - 0x1FFF	Reserved
0x2000	Module Register 0 (MR0) + Set Soft Lock Bit 0 (LMR0)
0x2001	Module Register 1 (MR1) + Set Soft Lock Bit 1 (LMR1)
0x2002 - 0x37FF	Module Register 2 (MR2) + Set Soft Lock Bit 2 (LMR2) - Module Register 6143 (MR6143) + Set Soft Lock Bit 6143 (LMR6143)
0x3800	Soft Lock Bit Register 0 (SLBR0): Soft Lock Bits 0-3
0x3801	Soft Lock Bit Register 1 (SLBR1): Soft Lock Bits 4-7
0x3802 - 0x3DFF	Soft Lock Bit Register 2 (SLBR2): Soft Lock Bits 8-11 -Soft Lock Bit Register 1535 (SLBR1535): Soft Lock Bits 6140-6143
0x3E00 - 0x3FFB	Reserved
0x3FFC	Global Configuration Register (GCR)

Note

Reserved registers in area #1 will be handled as specified in the protected module's documentation.

77.6.2 Register descriptions

This section describes in address order all the REG_PROT registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

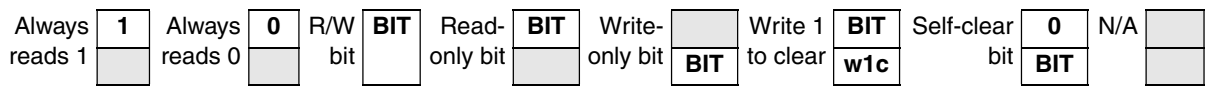


Figure 77-3. Register description

77.6.2.1 Module Registers (MR0-6143)

This is the lower 6 KB module memory space which holds all the functional registers of the module that is protected by the REG_PROT module.

77.6.2.2 Module register and set soft lock bit (LMR0-6143)

This is memory area #3 that provides mirrored access to the MR0-6143 registers with the side effect of setting Soft Lock Bits in case of a write access to a MR that is defined as protectable by the locking mechanism. Each MR is protectable by one associated bit in a SLBRn.SLBm, according to the mapping described in the Soft Lock Bit Register section.

77.7 Memory map and registers

REG_PROT memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
3800	Soft Lock Bit Register n (REG_PROT_SLBRn)	8	R/W	00h	77.7.1/3296
3FFC	Global Configuration Register (REG_PROT_GCR)	32	R/W	0000_0000h	77.7.2/3298

77.7.1 Soft Lock Bit Register *n* (REG_PROT_SLBRn)

The Soft Lock Bit Registers hold the Soft Lock Bits for the protected registers in memory area #1, which is the normal register address space of the protected module. Each SLB register has a four Soft Lock Bits (SLB0-SLB3), each of which controls write access to a byte in memory area #1. Each Soft Lock Bit also has a corresponding Write Enable bit in the same register that controls whether the Soft Lock Bit can be written. The following table shows the mapping between the Soft Lock Bits to the bytes in memory area #1.

Table 77-14. Soft Lock Bits vs. Protected Address

Soft Lock Bit	Protected address
SLBR0.SLB0	MR0
SLBR0.SLB1	MR1
SLBR0.SLB2	MR2
SLBR0.SLB3	MR3
SLBR1.SLB0	MR4
SLBR1.SLB1	MR5
SLBR1.SLB2	MR6
SLBR1.SLB3	MR7
SLBR2.SLB0	MR8
...	...

NOTE

- As many as 1536 Soft Lock Bit Registers (SLBRn) may be implemented, depending on the number of protected module register bytes.
- Access in User Mode is read-only; in Supervisor Mode access is read/write.

Address: 0h base + 3800h offset = 3800h

Bit	0	1	2	3	4	5	6	7
Read	0	0	0	0	SLB0	SLB1	SLB2	SLB3
Write	WE0	WE1	WE2	WE3				
Reset	0	0	0	0	0	0	0	0

REG_PROT_SLBRn field descriptions

Field	Description
0 WE0	Write Enable Bits for Soft Lock Bits (SLB) WE0 enables writing to SLB0

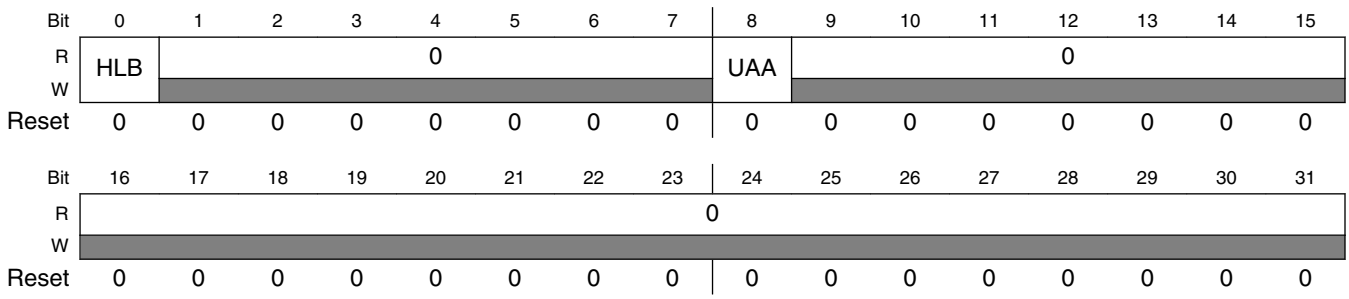
Table continues on the next page...

REG_PROT_SLBRn field descriptions (continued)

Field	Description
	1 Value is written to SLB 0 SLB is not modified
1 WE1	Write Enable Bits for Soft Lock Bits (SLB) WE1 enables writing to SLB1 1 Value is written to SLB 0 SLB is not modified
2 WE2	Write Enable Bits for Soft Lock Bits (SLB) WE2 enables writing to SLB2 1 Value is written to SLB 0 SLB is not modified
3 WE3	Write Enable Bits for Soft Lock Bits (SLB) WE3 enables writing to SLB3 1 Value is written to SLB 0 SLB is not modified
4 SLB0	Soft Lock Bits for one MRn register SLB0 can block accesses to MR[n *4 + 0] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
5 SLB1	Soft Lock Bits for one MRn register SLB1 can block accesses to MR[n *4 + 1] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
6 SLB2	Soft Lock Bits for one MRn register SLB2 can block accesses to MR[n *4 + 2] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable
7 SLB3	Soft Lock Bits for one MRn register SLB3 can block accesses to MR[n *4 + 3] 1 Associated MRn byte is locked against write accesses 0 Associated MRn byte is unprotected and writable

77.7.2 Global Configuration Register (REG_PROT_GCR)

Address: 0h base + 3FFCh offset = 3FFCh



REG_PROT_GCR field descriptions

Field	Description
0 HLB	Hard Lock Bit This register cannot be cleared once it is set by software. It can only be cleared by a system reset. **NOTE:** Access in User Mode is read/write if UAA is set to 1 otherwise it is read-only; in Supervisor Mode access is read/write. 1 All SLB bits are write protected and can not be modified. 0 All SLB bits are accessible and can be modified.
1–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8 UAA	User Access Allowed. Controls the User and Supervisor mode access to registers of the module under protection, as well as the REG_PROT_GCR and REG_PROT_SLBR*n*. 1 The registers in the module under protection can be accessed in the mode defined for the module registers without any additional restrictions. 0 The registers in the module under protection can be written in supervisor mode only. All write accesses in non-supervisor mode are not executed and a transfer error is issued. This access restriction is in addition to any access restrictions imposed by the protected IP module.
9–31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

77.8 Functional description

The following sections describe the functional characteristics of the Register Protection module.

77.8.1 General

This module provides a generic register (address) write-protection mechanism. The protection size can be:

- 32-bit (address == multiples of 4)
- 16-bit (address == multiples of 2)
- 8-bit (address == multiples of 1)

Which addresses are protected and the protection size depends on the SoC and/or module.

For all addresses (peripheral registers) that are protected there are SLBRn.SLBm bits that specify whether the address is locked. When an address is locked it can only be read but not written in any mode (supervisor/normal). If an address is unprotected the corresponding SLBRn.SLBm bit is always 0b0 no matter what software is writing to.

77.8.2 Change lock settings

To change the setting whether an address is locked or unlocked the corresponding SLBRn.SLBm bit needs to be changed. This can be done using the following methods:

- Modify the SLBRn.SLBm directly by writing to area #4
- Set the SLBRn.SLBm bit(s) by writing to the mirror module space (area #3)

Both methods are explained in the following sections.

77.8.2.1 Change lock settings directly via Area #4

In memory area #4 the lock bits are located. They can be modified by writing to them. Each SLBRn.SLBm bit has a mask bit SLBRn.WEm which protects it from being modified. This masking makes clear-modify-write operations unnecessary.

The following figure shows two modification examples. In the left example there is a write access to the SLBRn register specifying a mask value which allows modification of all SLBRn.SLBm bits. The example on the right specifies a mask which only allows modification of the bits SLBRn.SLB[3:1].

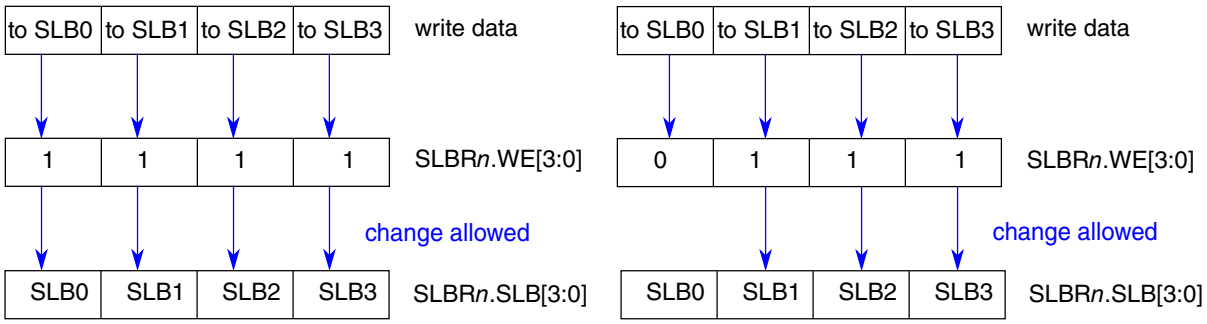


Figure 77-4. Change lock settings directly via Area #4

The previous figure shows four registers that can be protected 8-bit wise. The following figures show registers with 16- and 32-bit protection respectively:

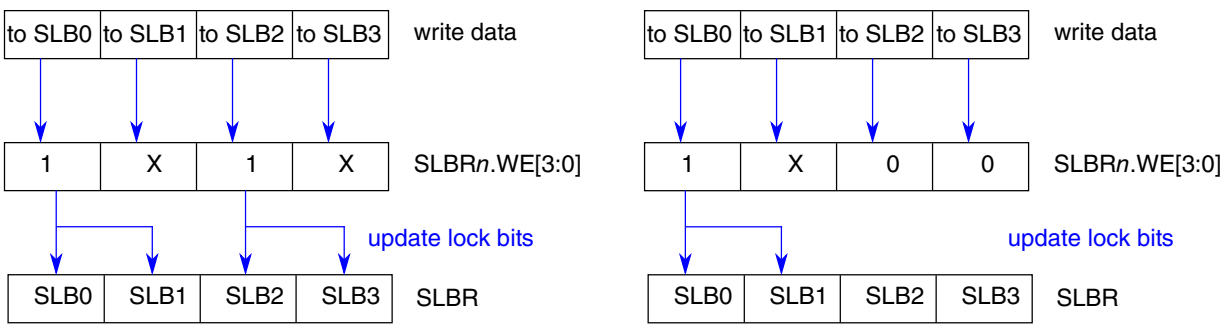


Figure 77-5. Change lock settings for 16-bit protected addresses

On the right side of the previous figure it is shown that the data written to SLBRn.SLB[0] is automatically written to SLBRn.SLB[1] also. This is done as the address reflected by SLBRn.SLB[0] is protected 16-bit wise. Note that in this case the write enable SLBRn.WE[0] must be set while SLBRn.WE[1] does not matter. As the enable bits SLBRn.WE[3:2] are cleared the lock bits SLBRn.SLB[3:2] remain unchanged.

In the example on the left side of the previous figure the data written to SLBRn.SLB[0] is mirrored to SLBRn.SLB[1] and the data written to SLBRn.SLB[2] is mirrored to SLBRn.SLB[3] as for both registers the write enables are set.

The next figure shows a 32-bit wise protected register. When SLBRn.WE[0] is set the data written to SLBRn.SLB[0] is automatically written to SLBRn.SLB[3:1] also. Otherwise SLBRn.SLB[3:0] remains unchanged.

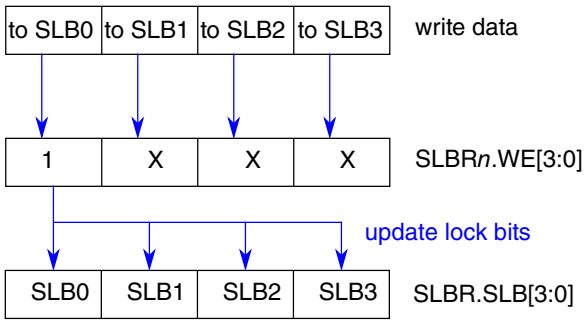


Figure 77-6. Change lock settings for 32-bit protected addresses

The following figure shows a mixed protection size configuration:

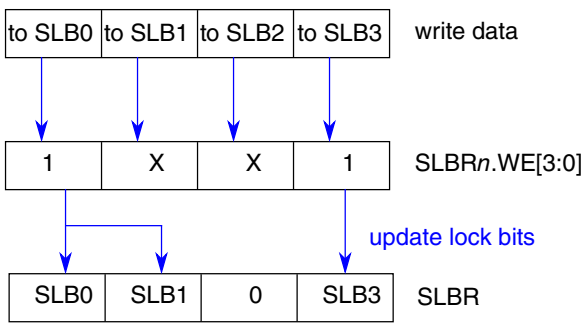


Figure 77-7. Change lock settings for mixed protection

The data written to SLBRn.SLB[0] is mirrored to SLBRn.SLB[1] as the corresponding register is 16-bit protected. The data written to SLBRn.SLB[2] is blocked as the corresponding register is unprotected. The data written to SLBRn.SLB[3] is written to SLBRn.SLB[3].

77.8.2.2 Enable locking via mirror module space (Area #3)

It is possible to enable locking for a register after writing to it. To do so the mirrored module address space must be used. For example:

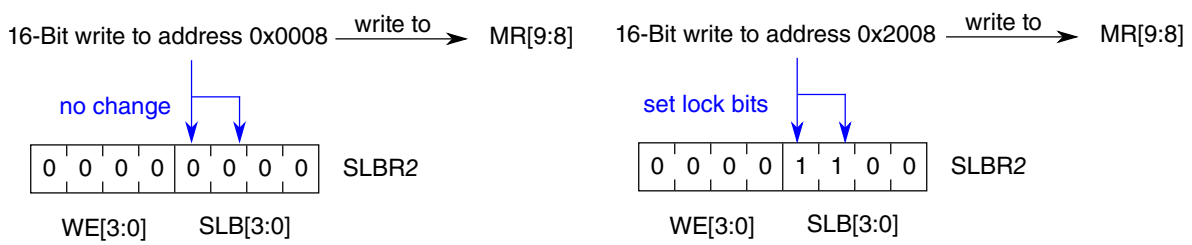


Figure 77-8. Enable Locking Via Mirror Module Space (Area #3)

When writing to address 0x0008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits remain unchanged (see the left part of Figure 77-5).

When writing to address 0x2008 the registers MR9 and MR8 in the protected module are updated. The corresponding lock bits SLBR2.SLB[1:0] are set while the lock bits SLBR2.SLB[3:2] remain unchanged (right part of Figure 77-5).

The following figure shows an example where some addresses are protected and some are not:

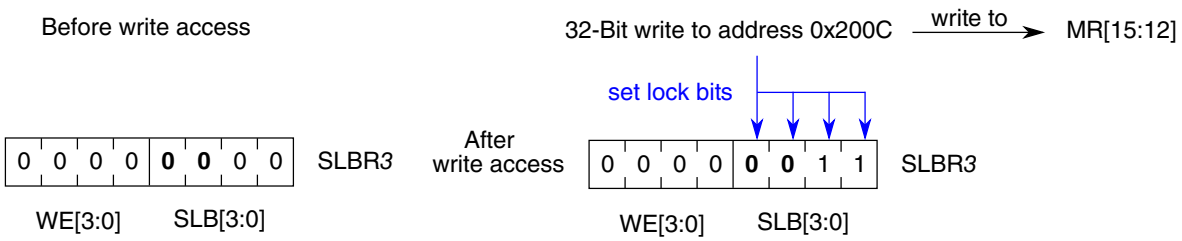


Figure 77-9. Enable locking for protected and unprotected addresses

In the previous figure addresses 0x0C and 0x0D are unprotected. Therefore their corresponding lock bits SLBR3.SLB[1:0] are always 0b0 (shown in bold). When doing a 32-bit write access to address 0x200C only lock bits SLBR3.SLB[3:2] are set while bits SLBR3.SLB[1:0] stay 0b0.

Note

Lock bits can only be set via writes to the mirror module space. Reads from the mirror module space will not change the lock bits.

77.8.2.3 Write protection for locking bits

Changing the locking bits through any of the procedures mentioned in Change lock settings directly via Area #4 and Enable locking via mirror module space (Area #3) is only possible as long as the bit GCR.HLB is cleared. Once this bit is set the locking bits can no longer be modified until there is a system reset.

77.8.3 Access errors

The protection module generates transfer errors under several circumstances. For the area definition refer to Figure 77-2

1. If accessing area #1 or area #3, the protection module will pass on any access error from the underlying Module under Protection.
2. If user mode is not allowed, user writes to all areas will assert a transfer error and the writes will be blocked.

3. If accessing the reserved area #2, a transfer error will be asserted.
4. If accessing unimplemented 32-bit registers in area #4 and area #5 a transfer error will be asserted.
5. If writing to a register in area #1 and area #3 with Soft Lock Bit set for any of the affected bytes a transfer error is asserted and the write will be blocked. Also the complete write operation to non-protected bytes in this word is ignored.
6. If writing to a Soft Lock Register in area #4 with the Hard Lock Bit being set a transfer error is asserted.
7. Any write operation in any access mode to area #3 while Hard Lock Bit GCR.HLB is set

Chapter 78
Password and Device Security Module (PASS)

78.1 Chip-specific PASS information

78.1.1 Password and Device Security Module (PASS) configuration

The PASS module is used to implement password-based read and write protection for flash blocks. Up to four levels of password protection can be implemented for each flash block.

NOTE

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

CAUTION

Do not access address offset 0x00D0 . This is for internal use only.

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

78.1.1.1 PASS_LOCK0_PGn register bit mapping

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Register Field	TSLOCK	Reserved	LOWLOCK[13:0]														MIDLOCK[15:0]															

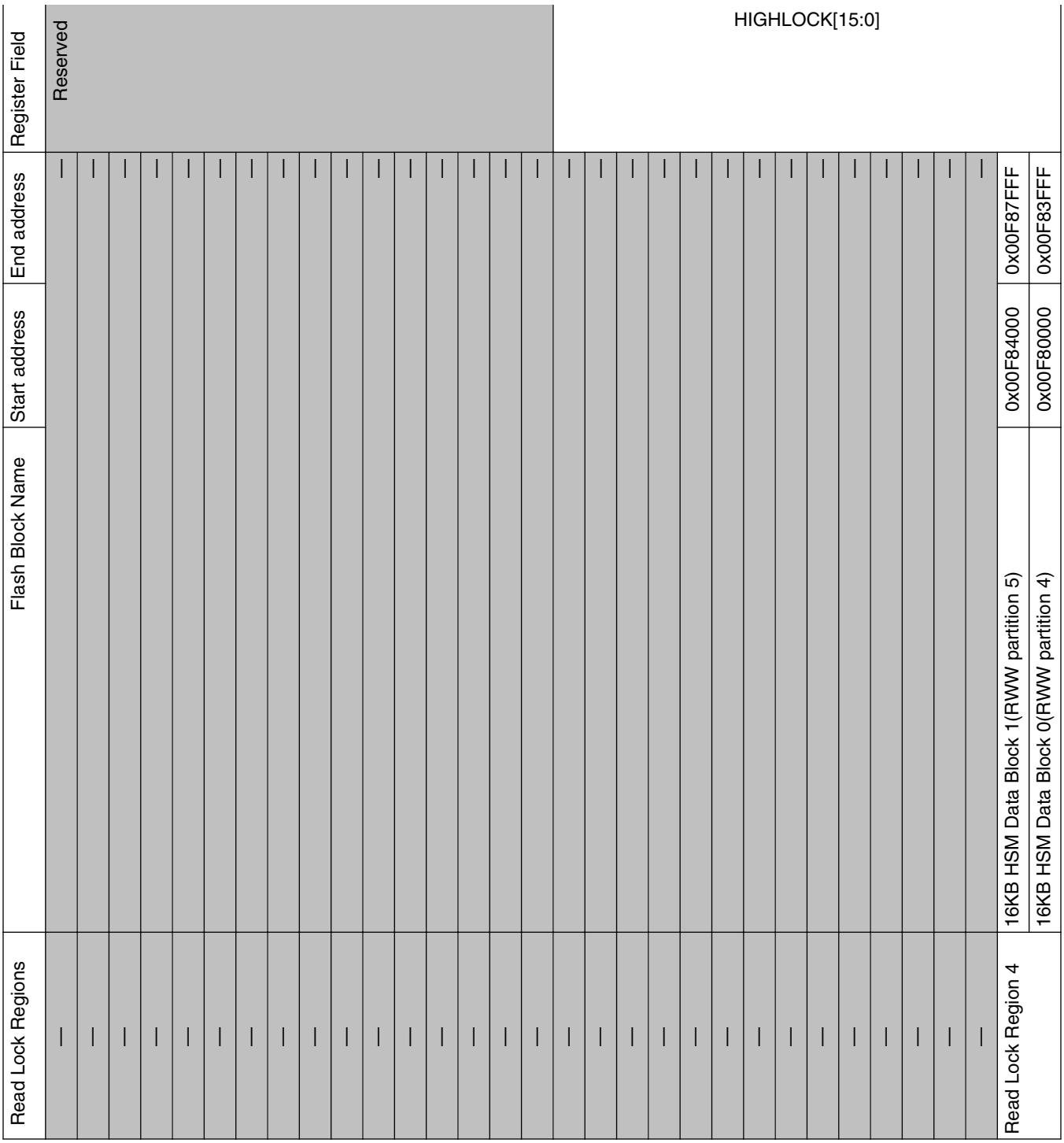
Table continues on the next page...

Read Lock Regions		Flash Block Name	Start address	End address
Read Lock Region 0	UTest NVM Block Space 16 KB		0x00400000	0x00403FFF
—				
—				
—				
—				
—				
Read Lock Region 3	64 KB HSMCode Block3 (RWW partition 1)		0x00620000	0x0062FFFF
Read Lock Region 1	64 KB Code Flash Block1 (RWW partition 1)		0x00FF0000	0x00FFFFFF
Read Lock Region 3	64 KB HSMCode Block2 (RWW partition 0)		0x00610000	0x0061FFFF
Read Lock Region 1	64 KB Code Flash Block0 (RWW partition 0)		0x00FE0000	0x00FEFFFF
	32 KB Code Flash Block3 (RWW partition 1)		0x00FD8000	0x00FDFFFF
	32 KB Code Flash Block2 (RWW partition 1)		0x00FD0000	0x00FD7FFF
	32 KB Code Flash Block1 (RWW partition 0)		0x00FC8000	0x00FCFFFF
	32 KB Code Flash Block0 (RWW partition 0)		0x00FC0000	0x00FC7FFF
—	16 KB BAF Block (RWW partition 1)		0x00404000	0x00407FFF
Read Lock Region 3	16 KB small HSM Code block		0x00F8C000	0x00F8FFFF
—				
—				
—				
—				
—				
—				
—				
—				
Read Lock Region 1	16 KB Code Flash Block 7 (RWW partition 3)		0x00FAC000	0x00FAFFFF
	16 KB Code Flash Block 6 (RWW partition 3)		0x00FA8000	0x00FABFFF
	16 KB Code Flash Block5 (RWW partition 3)		0x00FA4000	0x00FA7FFF
	16 KB Code Flash Block4 (RWW partition 3)		0x00FA0000	0x00FA3FFF
	16 KB Code Flash Block3 (RWW partition 2)		0x00F9C000	0x00F9FFFF
	16 KB Code Flash Block 2 (RWW partition 2)		0x00F98000	0x00F9BFFF
	16 KB Code Flash Block 1 (RWW partition 2)		0x00F94000	0x00F97FFF
	16 KB Code Flash Block 0 (RWW partition 2)		0x00F90000	0x00F93FFF

78.1.1.2 PASS_LOCK1_PGn register bit mapping

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table continues on the next page...



78.1.1.3 PASS_LOCK2_PGn register bit mapping

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table continues on the next page...

Read Lock Regions	Flash Block Name	Start address	End address	Register Field
—	256 KB Code Flash block9(RWW partition 7)	0x01240000	0x0127FFFF	L_256LCK
—	256 KB Code Flash block8(RWW partition 7)	0x01200000	0x0123FFFF	
—	256 KB Code Flash block7(RWW partition 6)	0x011C0000	0x011FFFFF	
—	256 KB Code Flash block6(RWW partition 6)	0x01180000	0x011BFFFF	
—	256 KB Code Flash block5(RWW partition 6)	0x01140000	0x0117FFFF	
—	256 KB Code Flash block4(RWW partition 6)	0x01100000	0x0113FFFF	
—	256 KB Code Flash block3(RWW partition 6)	0x010C0000	0x010FFFFF	
—	256 KB Code Flash block2(RWW partition 6)	0x01080000	0x010BFFFF	
—	256 KB Code Flash block1(RWW partition 6)	0x01040000	0x0107FFFF	
—	256 KB Code Flash block0(RWW partition 6)	0x01000000	0x0103FFFF	
Read Lock Region 1				

78.1.1.4 PASS_LOCK3_PGn register bit mapping

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table continues on the next page...

| Register Bit Name | PG | DBL | MO | Reserved | MSTR | Reserved | | | RL4 | RL3 | RL2 | RL1 | RL0 | U_256LCK | | |

Table 78-1. Debug Interface Enable Truth Table (continued)

Input Conditions				Outcome
Lifecycle	**Censorship**	**Debug Enable (LOCK3[DBL] bit)**	**JTAG Password**	**Debug Interface Enable**
	Censored	Unlocked (DBL=0)	Don't Care	Enabled
		Locked (DBL=1)	Matched	Enabled
			Not Matched	Blocked
Failure Analysis	Don't Care	Don't Care	Don't Care	Enabled

NOTE

User should keep external reset asserted to make the core enter debug mode from first instruction after JTAG Password Challenge.

78.1.1.6 Flash Test Mode protection

In Failure Analysis Lifecycle, Flash Test Mode entry is protected by combination of below two inputs.

1. DCF_FTMP (See)
2. TEST MODE ENABLE_DCF_CLIENT

Table 78-2. Flash Test Mode protection

DCF_FTMP	TEST MODE ENABLE DCF CLIENT	Flash TEST MODE ACCESS
0	Don't care	YES (NO Test mode enable PASSWORD REQUIRED)
1	0	Test Mode not enabled.
1	1	YES (Test mode enable PASSWORD REQUIRED)

78.2 Introduction

78.2.1 Overview

The Password and device security (PASS) module shown in the following figure, receives password challenges and determines their validity. It also maintains the device security and access states.

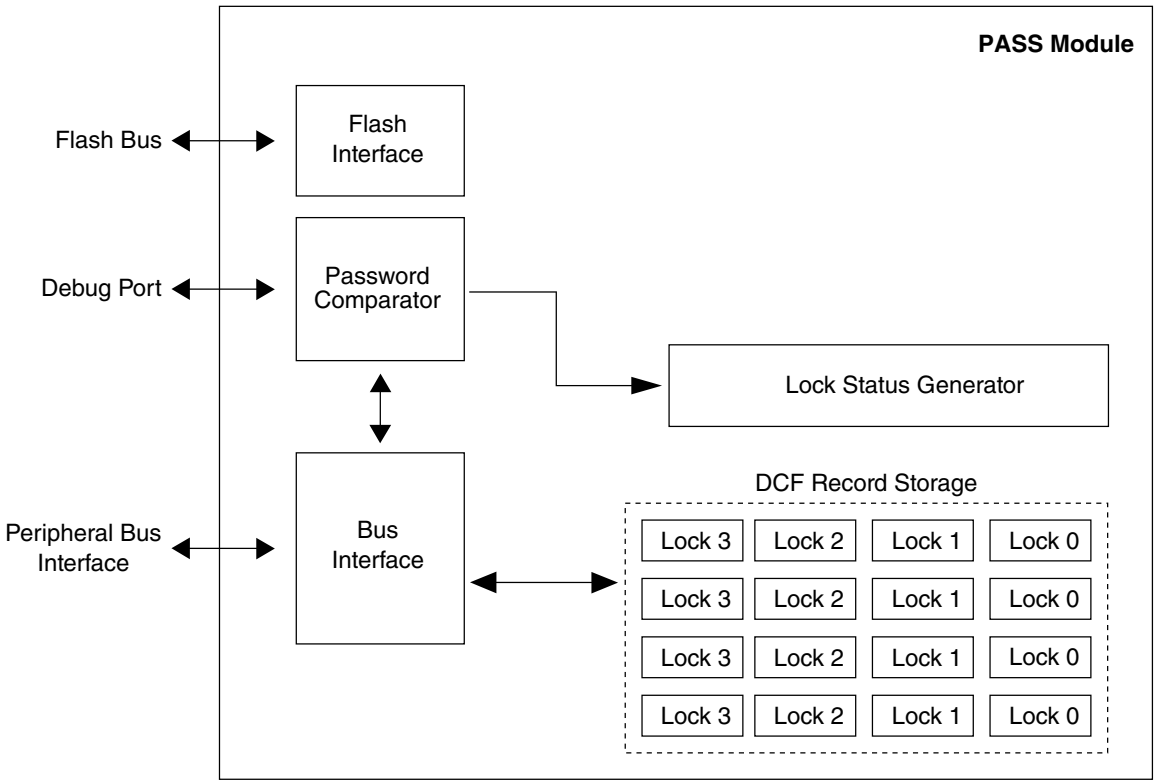


Figure 78-1. PASS module block diagram

78.2.2 Features

The PASS includes these distinctive features:

- Life Cycle status
- Password comparison
- JTAG password comparison
- Test mode enable Configuration and Password Comparison
- Flash read, write, and erase control

78.2.3 Modes of operation

The PASS operates identically in all system modes.

78.3 Memory map and register definition

This section provides a detailed description of all memory-mapped registers and DCF clients in the PASS.

Where a DCF client is listed, the reset value of that register may automatically be pre-loaded during reset based on the data stored in DCF Records in flash UTEST region.

The following table shows the memory map for the PASS. Note that all addresses are offsets; the absolute address for registers may be calculated by adding the specified offset to the base address of the PASS. The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section of the Reference Manual.

NOTE

Non-implemented registers generate bus aborts if enabled.

The following registers are available in the PASS. The shaded bits are reserved for future use. For future compatibility, only write '0' to these bits; they always read '0'.

PASS memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Life Cycle Status Register (PASS_LCSTAT)	32	R	See section	78.3.1/3313
8	Challenge Selector Register (PASS_CHSEL)	32	R/W	0000_0000h	78.3.2/3315
10	Challenge Status Register (PASS_CSTAT)	32	R	0000_0000h	78.3.3/3315
20	Challenge Input Register (PASS_CIN0)	32	W (always reads 0)	0000_0000h	78.3.4/3316
24	Challenge Input Register (PASS_CIN1)	32	W (always reads 0)	0000_0000h	78.3.4/3316
28	Challenge Input Register (PASS_CIN2)	32	W (always reads 0)	0000_0000h	78.3.4/3316
2C	Challenge Input Register (PASS_CIN3)	32	W (always reads 0)	0000_0000h	78.3.4/3316
30	Challenge Input Register (PASS_CIN4)	32	W (always reads 0)	0000_0000h	78.3.4/3316
34	Challenge Input Register (PASS_CIN5)	32	W (always reads 0)	0000_0000h	78.3.4/3316
38	Challenge Input Register (PASS_CIN6)	32	W (always reads 0)	0000_0000h	78.3.4/3316
3C	Challenge Input Register (PASS_CIN7)	32	W (always reads 0)	0000_0000h	78.3.4/3316

Table continues on the next page...

PASS memory map (continued)

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
100	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG0)	32	R/W	See section	78.3.5/3316
104	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG0)	32	R/W	See section	78.3.6/3318
108	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG0)	32	R/W	See section	78.3.7/3319
10C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG0)	32	R/W	See section	78.3.8/3320
110	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG1)	32	R/W	See section	78.3.5/3316
114	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG1)	32	R/W	See section	78.3.6/3318
118	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG1)	32	R/W	See section	78.3.7/3319
11C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG1)	32	R/W	See section	78.3.8/3320
120	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG2)	32	R/W	See section	78.3.5/3316
124	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG2)	32	R/W	See section	78.3.6/3318
128	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG2)	32	R/W	See section	78.3.7/3319
12C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG2)	32	R/W	See section	78.3.8/3320
130	Password Group n - Lock 0 Status Register (PASS_LOCK0_PG3)	32	R/W	See section	78.3.5/3316
134	Password Group n - Lock 1 Status Register (PASS_LOCK1_PG3)	32	R/W	See section	78.3.6/3318
138	Password Group n - Lock 2 Status Register (PASS_LOCK2_PG3)	32	R/W	See section	78.3.7/3319
13C	Password Group n - Lock 3 Status Register (PASS_LOCK3_PG3)	32	R/W	See section	78.3.8/3320

78.3.1 Life Cycle Status Register (PASS_LCSTAT)

The Life Cycle Status Register reflects the production status and DCF Censorship client status of the device. It is possible to mature the device (move the device life cycle forward), but never to revert the life cycle to an earlier state.

The reset value of the LCSTAT register LIFE field derives from an external life cycle management module (like SSCM). It is a 3-bit life cycle. This register simply displays the life cycle input from external blocks.

Address: 0h base + 0h offset = 0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CNS	JUN	TME	0												
W																
Reset	*	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0													LIFE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	*	*	*

* Notes:
- LIFE field: Reset value is the Life Cycle of the device, and is unaffected by reset.
- CNS field: Reset value depends on the Censorship DCF client

PASS_LCSTAT field descriptions

Field	Description
0 CNS	This field indicates whether the DCF Censorship client has been written with a DCF record to 0x55AA or if the DCF Censorship client contains any other value than 0x55AA. The bit can be set (or cleared) via DCF Record in UTest Flash. **NOTE:** If no DCF record has been written to affect the value of the CNS bit, its reset value will be 1. 1 DCF Censorship Client value is anything other than 0x55AA. 0 DCF Censorship client value is 0x55AA.
1 JUN	This field indicates whether the device has been temporarily uncensored. Successful transmission of the JTAG password sets this bit to '1', otherwise it will be 0. (In life cycle FA, the JUN bit will always be 0.) Unsuccessful transmission clears this bit.
2 TME	Test Mode enable. This field indicates whether the test mode DCF Client is set to enable test mode in FA. 0 Test mode cannot be enabled 1 Test mode can be enabled with test mode password
3–28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29–31 LIFE	The life cycle setting is manipulated externally (SSCM). Pass displays a 3-bit input life cycle 000 Failure Analysis 001 Reserved 010 OEM Production 011 Customer Delivery 100 Reserved 101 Reserved 110 MCU Production 111 In Field

78.3.2 Challenge Selector Register (PASS_CHSEL)

This register determines which password group is challenged via the Challenge Input Register.

When the Master Only (MO) bit of the PASS_LOCK3_PGn register is set, only the master that unlocked access to the passgroup settings will be able to access these registers. The CMST field of the Challenge Status Register will reflect the locking master. When the Master Only (MO) bit of the PASS_LOCK3_PGn register is not set, the passgroup settings can be accessed by any master.

When a master writes this register, its ID sets the CMST field of the Challenge Status register.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																														GRP	
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PASS_CHSEL field descriptions

Field	Description
0–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–31 GRP	Password Group to unlock 00 Password Group 0 01 Password Group 1 10 Password Group 2 11 Password Group 3

78.3.3 Challenge Status Register (PASS_CSTAT)

This register provides status information for the password challenge.

Only the master shown in this register can access the CINn registers - access by other masters result in a comparison fail regardless of the values written.

Address: 0h base + 10h offset = 10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																												CMST			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PASS_CSTAT field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–31 CMST	ID of the master which has last written the CHSEL register

78.3.4 Challenge Input Register (PASS_CIN*n*)

This register is used to provide the Challenge word *n* for a password group as selected by the Challenge Selector Register.

After programming the Challenge Selector Register, the password challenge needs to be written with eight 32-bit writes from CIN0 to CIN7 in that order. The most significant 32 bits of the password challenge must be written to CIN0. Once CIN7 is written, the password is compared. The write access only completes after the comparison has been executed. In case of a successful password comparison, the PASS module stores the ID of the master which has written CIN0 to CIN7 in the PASS_LOCK3_PGn register of the selected password group, as well as clears the PGL bit of that register.

All accesses to these registers need to be made by the bus master that wrote the CHSEL register previously. If another master writes any of the CIN registers before the original master has written CIN7, the comparison fails.

Address: 0h base + 20h offset + (4d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	PW32																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PASS_CIN*n* field descriptions

Field	Description
0–31 PW32	32 bits of the 256-bit password challenge. Only 32-bit writes may be used.

78.3.5 Password Group *n* - Lock 0 Status Register (PASS_LOCK0_PG*n*)

Each Password Group has one Lock 0 Status register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked the register becomes writable. For additional details of the LOCK bits, refer to the Flash memory description.

The LOCK bits do not come into effect before the life cycle has matured to *OEM Production* or older.

LOCK0_PG*n* register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset. The default value before any DCF record is written is '1', indicating that a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes to this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is OEM Production or older, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 100h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TSLOCK	ATSL	LOWLOCK													
W																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIDLOCK															
W																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- MIDLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to ‘1’.

- LOWLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to ‘1’.
- ATSL field: If no DCF records have been written to change the ATSL bit state, the bit will reset to ‘1’.
- TSLOCK field: If no DCF records have been written to change the TS bit state, the bit will reset to ‘1’.

PASS_LOCK0_PG*n* field descriptions

Field	Description
0 TSLOCK	UTest NVM Lock. This bit is used to lock the UTest NVM block from programs (erase protection not needed since UTest NVM is OTP and not erasable).
1 ATSL	Alternate Interface UTest NVM Lock. This bit is used to lock the UTest NVM block from programs on the alternate interface (erase protection not needed since UTest NVM is OTP and not erasable).
2–15 LOWLOCK	Low Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.
16–31 MIDLOCK	Mid Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

78.3.6 Password Group *n* - Lock 1 Status Register (PASS_LOCK1_PG*n*)

Each Password Group has one Lock 1 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable . For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the life cycle matures to OEM Production or older.

LOCK1_PG*n* register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset . The default value before any DCF record is written is '1', which indicates a block is locked. If no DCF records have been written to change the lock state, the bits will reset to '1'.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group to prevent permanent and irreversible locking of Flash blocks.

CAUTION

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF

Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is older than OEM Production, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 104h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0																HIGHLOCK															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:
- HIGHLOCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to ‘1’.

PASS_LOCK1_PG*n* field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 HIGHLOCK	High Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

78.3.7 Password Group *n* - Lock 2 Status Register (PASS_LOCK2_PG*n*)

Each Password Group has one Lock 2 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable . For additional details of the LOCK bits, refer to the Flash description.

The LOCK bits do not come into effect before the life cycle is matured to OEM Production or older.

NOTE

Lock registers can only be accessed with 32-bit operations.

LOCK2_PG*n* register functions are as shown in this section.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset. The default value before any DCF record is written is '1', which indicates a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is older than OEM Production, the Flash blocks are always unlocked, regardless of the values in the password groups.

Address: 0h base + 108h offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	L_256LCK																															
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- L_256LCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit resets to '1'.

PASS_LOCK2_PG*n* field descriptions

Field	Description
0–31 L_256LCK	Lower 256KByte Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

78.3.8 Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*)

Each Password Group has one Lock 3 Status Register. This register shows the current lock state for the Flash blocks. Each time a password group is unlocked, the register becomes writable.

The LOCK bits do not come into effect before the life cycle matures to OEM Production or older.

LOCK3_PGn register functions are as shown in this section.

NOTE

Lock registers can only be accessed with 32-bit operations.

The initial value of the lock bits is set via DCF records stored in UTEST Flash block and copied over during functional reset. The default value before any DCF record is written is '1', which indicates a block is locked.

During the Customer Delivery life cycle, the user must program a valid password or a DCF record that writes this register to unprotect the UTEST for each password group in order to prevent permanent and irreversible locking of Flash blocks.

Warning

Failure to program valid passwords or DCF records for all password groups before the life cycle matures to OEM Production renders the part unusable. However, additional DCF Records can be written to UTEST to change the initial value of this register.

Once a password group is unlocked, software can set or clear any of the lock bits in the group, by writing to the registers.

The resulting lock status of a Flash block is determined by the combination of the lock bits of all password groups. If a block is locked in multiple groups, then all lock bits for the block need to be cleared before program and erase is possible. However, unless the life cycle is older than OEM Production, the Flash blocks are always unlocked, regardless of the values in the password groups.

Once a censored device enters the Failure Analysis life cycle, read protection is active regardless whether the system is in debug mode or not. It is still possible to unlock Flash regions with the correct passwords, however. Flash Utest region controlled by LOCK3[0] is read unlocked in Failure Analysis life cycle.

NOTE

Do not toggle DBL and RLx bits with the same write command. Use one instruction to program the DBL bit and a separate instruction to toggle the RLx bits.

Address: 0h base + 10Ch offset + (16d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0			RL4	RL3	RL2	RL1	RL0
W																
Reset	*	*	*	0	1	1	1	1	0	0	0	*	*	*	*	*

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	U_256LCK															
W																
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Notes:

- U_256LCK field: For each implemented bit of this field, if no DCF records have been written to change the lock state, the bit will reset to '1'.
- RL0 field: If no DCF records have been written to change the RL0 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL1 field: If no DCF records have been written to change the RL1 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL2 field: If no DCF records have been written to change the RL2 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL3 field: If no DCF records have been written to change the RL3 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- RL4 field: If no DCF records have been written to change the RL4 bit lock state, the bit will reset to a logic 1. This Flash block region is always readable if the life cycle is Customer Delivery or earlier. To determine if this region is readable for Life Cycle states later than Customer Delivery, consult the Flash Memory Read Protection Truth Table.
- MO field: If no DCF records have been written to change the MO bit lock state, the bit will reset to ‘1’.
- DBL field: If no DCF records have been written to change the DBL bit lock state, the bit will reset to ‘1’.
- PGL field: Reset value depends on life cycle. If life cycle is Customer Delivery or earlier, the value will be 0, otherwise 1.

PASS_LOCK3_PG*n* field descriptions

Field	Description
0 PGL	Password Group Lock. This bit may be set in software to lock the password group, when the device is older than Customer Delivery. Before that the value of the bit is always 0. This bit is only updatable from SW path. It cannot be updated from DCF load from Flash. This bit can be cleared by writing a valid password to the Password Challenge Input Registers. 0 Password group registers are unlocked. All four registers in the Password group may be read and written without restriction. 1 Password group registers are locked. Writes to the password group registers have no effect. Read accesses work normally. (If life cycle is Customer Delivery or earlier, this bit has no effect and the registers are always unlocked.)

Table continues on the next page...

PASS_LOCK3_PG*n* field descriptions (continued)

Field	Description
1 DBL	Debug Interface Lock. The default value of this bit would be '1' if not updated from flash. 0 Debug interface is unlocked. Accessed to all JTAG clients is allowed (Except HSM) 1 Debug interface is locked. Only access to JTAG password challenge register is possible. Access to all other JTAG clients is blocked
2 MO	Master Only. 0 All masters can modify the passgroup settings if PGL is cleared 1 Only the Master which unlocked the passgroup can change the passgroup settings
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–7 MSTR	Master Access. Only the Master with this matching master ID can change the passgroup settings. This field is loaded with the ID of the master which unlocked the passgroup once the correct password is provided.
8–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 RL4	Read Lock Regison 4 lock bit. The default value of this bit would be '1' if not updated from flash. 0 Region 4 is not protected. 1 Region 4 is protected.
12 RL3	Region 3 is protected. The default value of this bit would be '1' if not updated from flash. 0 Region 3 is not protected. 1 Region 3 is protected.
13 RL2	Region 2 is protected. The default value of this bit would be '1' if not updated from flash. 0 Region 2 is not protected. 1 Region 2 is protected.
14 RL1	Region 1 is protected. The default value of this bit would be '1' if not updated from flash. 0 Region 1 is not protected. 1 Region 1 is protected.
15 RL0	Region 0 is protected. The default value of this bit would be '1' if not updated from flash. 0 Region 0 is not protected. 1 Region 0 is protected.
16–31 U_256LCK	Upper 256 KB Block Lock. A value of 1 in a bit of the lock register signifies that the corresponding block is locked for program and erase.

78.4 DCF clients

The following table shows the DCF clients for the PASS.

Where a DCF client is listed, the reset value of that register may automatically be pre-loaded during reset based on the data stored in DCF Records in Flash UTEST region.

The address of the DCF client is local to the PASS module and must be qualified with the appropriate Client Select (CS[14:0]) for the PASS module, as defined in the DCF Record Section of the Reference Manual or in DCF sheet attached to this document.

Table 78-3. PASS DCF clients

Offset	Register	Access	Reset Value	DCF Client Address (ADDR[14:0])	Section
Censorship					
None	Censorship	None	0x0000_0000	0x00B0	Censorship
Test mode enable					
None	Test mode enable	None	0x0000_0001	0x00E0	Test mode enable DCF Client

78.4.1 Censorship

Censorship is a DCF client that is *not* accessible by software. It is automatically pre-loaded during reset with data stored in DCF Record format in the Flash UTEST region.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 This bit is not defined

2 This value is loaded by DCF Record during reset

Field	Description
16:31	Censorship If this value is 0x55AA the MCU is NOT censored If this value is anything other than 0x55AA, the MCU is censored

78.4.2 Test mode enable DCF Client

This DCF client is used to configure the password protected Test mode enable feature. See Test mode enable for details.

The Test mode DCF Client is not directly readable in software, but the state of the Test mode bit can be read in the LCSTAT register (Life Cycle Status Register (PASS_LCSTAT)).

DCF Client Address: 0x00E0 Test Mode Enable — Access:DCF Record Read Once Only

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R																
W																
Reset	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																
W																TM[2]
Reset	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	x[1]	1

1 This bit is not defined

2 This value is loaded by DCF Record during reset

Figure 78-2. Test mode enable DCF Client

Table 78-4. Test mode enable register field descriptions

Field	Description
0-30	Reserved
31 TM	Test mode enable '0' = Test mode cannot be enabled '1' = Test mode can be enabled with a password

78.5 Functional description

The primary purpose of the PASS is to provide fine-grained access control to the Flash and debugger interface. It utilizes the DCF mechanism, as described in the SSCM section.

78.5.1 Censorship

Censorship is the overall control that enables and disables two independent security features: Debug Interface access and flash memory read protection.

The Debug Interface access is controlled by the CNS and LIFE bits of the LCSTAT register (Life Cycle Status Register (PASS_LCSTAT)) according to the Debug Interface Access truth table. The flash memory read protection is also controlled by the CNS and LIFE bits of the LCSTAT register according to the Flash memory read protection truth table.

The CNS bit is set or cleared by executing a DCF record that writes to the Censorship DCF client (See Censorship). The LIFE bits of the LCSTAT register are controlled by the SSCM using user programmed values from specific locations in the UTEST flash.

78.5.2 Debug Interface Access

The debug interface is fully enabled in Customer Delivery life cycle. But from OEM Production life cycle, it can be blocked using censorship and unlocked by either entering a 256 bits password via JTAG or by software clearing the DBL bit the PASS_LOCK3 registers (See Password Group *n* - Lock 3 Status Register (PASS_LOCK3_PG*n*)). Both these debug enabling options are further conditioned by life cycle.

The debug interface enable truth table for this chip is shown in the device specific PASS section of this chapter.

78.5.2.1 JTAG Password Challenge

To enable the Debug Interface Access, the JTAG password challenge register in the JTAGC, can be written by an external tool. If that password matches the JTAG password stored in UTEST flash, Debug Interface Access is enabled. The password matching is done only after the chip has come out of reset PHASE3. The Debug Interface Access stays enabled until the next destructive reset.

78.5.3 Flash Memory Read Protection

Flash Memory Read Protection is a security feature that can selectively make regions of the flash memory unreadable. Any attempt to read a region of flash that is read protected results in the read access terminating with a transfer error. This is consistent for all bus masters including debug bus masters.

The assertion of flash memory region read protection is controlled by Life Cycle, Censorship and whether the Debug Interface access is enabled or blocked. The Flash memory read protection truth table describes the combination of conditions that drive Flash Memory Read Protection.

The intention of the feature is to protect the contents of the flash from being read. The flash memory is readable in normal operation, allowing the application to run, but becomes read protected when a Debug tool is attached and enabled (JTAG password or LOCK3[DBL] bit).

The flash memory is also read protected when the life cycle status is Failure Analysis. This life cycle is used for a module field failure, where the device is returned for failure analysis. This protection prevents the contents of the flash memory from being read.

The flash memory is divided into a number of Regions (3 to 5 depending on the device). The assignment of the regions is defined in the device reference manual.

Table 78-5. Flash Memory Read Protection Truth Table

INPUTS				OUTPUTS	
Life Cycle	**Censorship**	**Debug Interface Enable from HSM**	**Lock3[RLx]**	**Read Protected RL[0] (UTEST)**	**Read Protected RL[4:1] (Other Regions)**
Customer Delivery	Don't Care	Don't Care	Don't Care	Readable	Readable
OEM Production / IN Field	Uncensored	Don't Care	Don't Care	Readable	Readable
	Censored	Blocked	Don't Care	Readable	Readable
		Enabled	Locked (1)	Read Blocked	Read Blocked
			Unlocked (0)	Readable	Readable

Table continues on the next page...

Table 78-5. Flash Memory Read Protection Truth Table (continued)

INPUTS				OUTPUTS	
Life Cycle	**Censorship**	**Debug Interface Enable from HSM**	**Lock3[RLx]**	**Read Protected RL[0] (UTEST)**	**Read Protected RL[4:1] (Other Regions)**
Failure Analysis	Uncensored	Don't Care	Don't Care	Readable	Readable
	Censored	Don't Care	Locked (1)	Readable	Read Blocked
			Unlocked (0)	Readable	Readable

78.5.4 Flash Memory Write and Erase Protection

This section describes write and erase permissions for flash memory. From Power ON to SSCM Done, all regions of flash memory are in locked state. After Power ON, SSCM scans flash memory for valid DCF records. These DCF records carry program and erase protection settings for flash memory regions. SSCM module finds the settings and load them into PASS configuration via the DCF path. Once SSCM Done is asserted, the new configuration loaded by SSCM into PASS come into force and unlocks different regions of flash for program and erase.

To enable write and erase access of a particular region of flash memory, add corresponding DCF records in the flash memory once chip comes out of reset (i.e. SSCM Done). User can also change the settings by providing valid password to the PASS module, and by modifying the settings of following registers bits fields:

- PASS_LOCK0_PGn[MIDLOCK]
- PASS_LOCK0_PGn[LOWLOCK]
- PASS_LOCK1_PGn[HIGHLOCK]
- PASS_LOCK2_PGn[L_256LCK]
- PASS_LOCK3_PGn[U_256LCK]

See chip-specific PASS information for mapping of the above mentioned bits fields.

In the PASS module there is no individual access protection for Write or Erase access types, but Write and Erase accesses summated.

78.5.5 Test mode enable

Test mode can be used for testing different features of Flash and internal blocks. These features can be used in Failure analysis life cycle. Following are the guidelines for customers:

- During customer assembly, customer programmes DCF Record in UTEST to write test mode DCF client with 1.
- During customer assembly, the customer programmes the test mode enable Password into Flash
- The customer maintains a system for retrieving these passwords for the life of the module
- If the part is returned to JDP for FA, the customer must provide the test mode enable password that was originally programmed into the device.
- If the password is lost or irretrievable, the customer must accept that FA cannot be done.

NOTE

Usage of this case causes restrictions on failure analysis.

78.5.6 Test mode enable password

The Test mode enable password is a password entered by the manufacturer via JTAG interface, during device FA, to gain access to Flash Test Mode. Flash Test Mode is required to perform Vt Distribution analysis and other tests on an MCU. This password is only functional during FA life cycle. During other life cycles, Flash Test Mode is either available; completely blocked; or requires the Retest Password. Flash blocks that are assigned as sealed during production are locked against any form of Flash test, including reading, erasing and Vt Distribution analysis.

78.5.6.1 Test mode enable password generation

The Test mode enable password is created by the customer during their module assembly process. It must be programmed into UTEST Flash while in Customer Delivery life cycle.

When an MCU is returned to manufacturer for FA, it must be in FA life cycle and the password must also be provided to the manufacturer to allow access to Flash Test Mode. Therefore the customer should develop a process for managing Test mode enable passwords for the life of the module.

78.5.6.2 Test mode enable password storage

The test mode enable password is 224 bits (28 bytes) wide and is to be stored in UTEST Flash at a fixed location. See the device reference manual for the address of the test mode password.

The password region of UTEST Flash is readable only when in Customer Delivery life cycle. In any life cycle after this (including FA life cycle), this region is read protected.

78.5.6.3 Test mode enable password Challenge entry

The Test mode enable password challenge in entered by an external tool to the password register in JTAGC

As the Test mode enable password is 224 bits; and the JTAGC password register is 256, only the least significant 224 bits of the register are used.

78.5.6.4 Test mode enable password compare

The Test mode password challenged, is compared against the password stored in UTEST Flash. Only the least significant 224 bits are compared as the most significant 32 bits of the Flash page contain different data not related to Test mode enable.

If the password matches, Flash Test Mode access is allowed until the next POR reset.

There are no passwords that the hardware will detect as illegal. All 0 and all 1 are valid passwords.

78.6 Initialization/application information

78.6.1 Reset

The reset state of each individual bit is shown within the Register Description section. However, many bits receive their values based on DCF records stored in the UTEST Flash memory block.

78.6.2 Setting lock bits in a password group

There is no restriction on setting and clearing lock registers in the UTEST Flash, apart from the available space for DCF records. It is possible to set and clear them multiple times.

Table 78-6. CS/ADDR Values for Lock Registers

Register	DCF Command Word
PW Group 0	
Lock0	0xxxxx[1] _0100
Lock1	0xxxxx[1] _0104
Lock2	0xxxxx[1] _0108
Lock3	0xxxxx[1] _010C
PW Group 1	
Lock0	0xxxxx[1] _0110
Lock1	0xxxxx[1] _0114
Lock2	0xxxxx[1] _0118
Lock3	0xxxxx[1] _011C
PW Group 2	
Lock0	0xxxxx[1] _0120
Lock1	0xxxxx[1] _0124
Lock2	0xxxxx[1] _0128
Lock3	0xxxxx[1] _012C
PW Group3	
Lock0	0xxxxx[1] _0130
Lock1	0xxxxx[1] _0134
Lock2	0xxxxx[1] _0138
Lock3	0xxxxx[1] _013C

1. Value depends on the DCF Client Select assigned to the PASS module in the device. See chip specific PASS information for the Client Select assignment

In order to set the MIDLOCK field in the Lock0 register of password group 1 to the value 0x00FF and other fields to 0, the DCF record has to be added as shown in Table 78-7.

Table 78-7. Setting some bits in the MIDLOCK field

ADDR offset	DATA	Description
0x00	0x05AA_55AF	Start Record
0x04	0x0000_0000	
0x08	0x0000_00FF	PW Group 1, Lock0
0x0C	0xxxxx_0110	Set MIDLOCK to 0xFF
0x10	0xFFFF_FFFF	Non-Programmed Addresses
0x14	0xFFFF_FFFF	
…	…	

Chapter 79
Tamper Detect Module (TDM)

79.1 Chip-specific TDM information

79.1.1 Tamper Detection Module (TDM) configuration

The Tamper Detection Module provides a type of flash memory write protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. An additional feature is provided to enable customers to override the protection mechanism for any TDR so that no record is required to be written before a block within the TDR can be erased.

Additionally, the TDM provides a mechanism for configuring individual flash memory blocks as One Time Programmable (OTP), i.e., write once, blocks.

The following sections provide details on the TDM configuration registers and DCF clients.

NOTE

The TDM configuration registers detailed in this section are DCF clients. In general, DCF clients can only be written once —their intended values must be carefully selected beforehand.

79.1.2 Diary Base Address (DBA) DCF client

DBA DCF client holds the base address of the diary, which is the region of flash that contains records corresponding to erase operations.

79.1.3 Tamper Region Override (TO) DCF client

Tamper detection can be overridden by writing to enable bits (one per region) in the Tamper Region Override (TO) DCF client. The mapping is shown below.

DCF Client Address: 0x0001 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0
W																																
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The mapping of the Tamper Region Override Enable (TOEn) DCF client is as follows:

Table 79-1. TOEn field mapping

Field	TDR
TOE0	0
TOE1	1
TOE2	2
TOE3	3
TOE4	4
TOE5	5

79.1.4 OTPEN*n* DCF clients

Configuring individual flash blocks as OTP blocks is accomplished by setting the appropriate values in the TDM_OTPENn DCF clients. Each mapped bit is associated with a specific flash block. A “1” bit value indicates a block is OTP.

The mappings are shown in the attached TDR and OTP LOCK Register excel .

79.1.5 TDRx_LOCKn DCF clients

Configuration of the 6 tamper regions is accomplished by setting field values in LOCK DCF clients (4 for each TDR).

Each set of 4 LOCKn DCF clients acts as a map of the MCU's flash memory blocks. A "1" in a LOCKn bit indicates that software must write a record before the corresponding block can be modified. The mapping of TDM_LOCKn DCF client bits to flash blocks is identical to the OTPENn mapping as shown in the sections above.

Any available flash block can be mapped to any tamper protection region.

79.1.6 TDM register reset value

All register reset values for the TDM registers that are device specific are shown below. All other registers reset values are shown in the TDM Memory chapter.

TDR Status Register (TDM_TDRSR) resets to 0x0000_ 003Fh.

79.2 Overview

The Tamper Detection Module provides a type of flash memory erase protection mechanism that requires software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. The mechanism does not *prohibit* an erase operation on any block within a TDR—it requires a record to be written before the operation can execute. Collectively, the records are referred to as a "diary". At minimum, the diary serves as an erase counter. Because the diary record content is customer-defined, the diary can also serve as a history.

Up to 6 TDRs can be defined via DCF records.

The following figure shows the block diagram of the TDM and its interface to other system components.

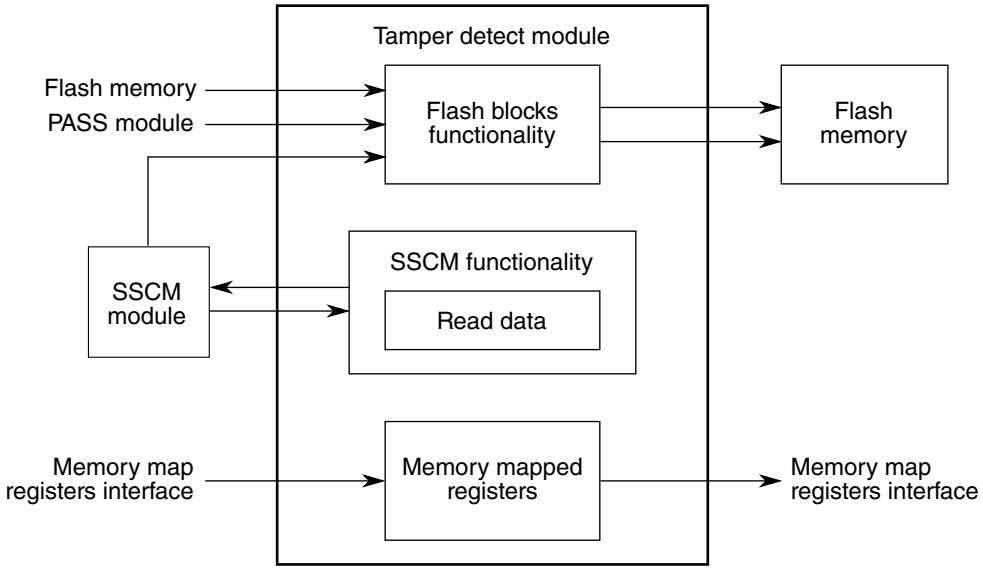


Figure 79-1. Tamper Detection Module block diagram

Security information is passed to the TDM by the SSCM module via the DCF bus. Among the data are:

- OTP (One Time Programmable) settings for each of the flash blocks
- Base address for the diary in the flash
- Block assignments for the TDRs
- Tamper region override information
- Software tamper region disable

The TDM module controls the flash blocks for erase operations and sets them as OTP.

The two flash buses are used to control the flash blocks for erase operation and set them as OTP. OTP means that flash erase of the entire block is disabled and the only words that are already erased can be programmed. Over-programming is not possible.

NOTE

The pass module provides the control for erase operation for each flash block as well as for each region of flash memory, and the PASS settings have priority over the settings of TDM.

The flash block indicates to the TDM the address of the programmed word. This address is internally decoded and provides a means to verify whether the block is part of any TDRs. The memory map registers interface provides a means to the user to read information of the TDM registers.

79.3 Features

The Tamper Detection Module supports these distinctive features:

- Erase counter (diary) for each TDR to record up to 256 write attempts on the flash blocks
  - The diary is divided into 6 parts (TDRs), each consisting of 256 64-bit records.
  - The diary block should be assigned as OTP region in flash.
  - The diary area can store any type of relevant information.
- Tamper Detection Region Block Assignment
  - Any flash block can be assigned to any TDR and be controlled.
- 32-bit Erase Lock DCF clients for each flash memory TDR defined.
- 32-bit OTP Enable DCF clients to disable the erase operation for an entire flash block
- Override TDR via DCF records[1]
  - Allows block assigned to a TDR to be unlocked for erase even if the TDR is locked for this operation.
- Memory mapped registers interface for reading via software of specific registers.

79.4 External signal description

There are no external signals driving or being driven by the TDM.

79.5 DCF clients

TDM-related DCF records perform the following functions:

- Establish a permanent diary base address
- Define Tamper Detect Regions (TDRs) to monitor on-chip flash memory program/ erase activity
- Permanently disable one or more Tamper Regions
- Configure flash memory as One Time Programmable (OTP) on a per-block basis

The following table gives an overview on the TDM DCF clients implemented.

1. The Tamper Region Override DCF record is a "write one only" DCF record. If an override is applied to a TDR it is permanent and cannot be reversed.

Table 79-2. DCF clients

DCF Client Name	Description	Strategy
Diary Base Address	Stores the base address of the tamper detection diary	write once, write 1 to 0 only
Tamper Region Override	Stores information of which TDR is to be overridden	write 0 to 1 only
Software Tamper Region Override Disable	Stores information to disable the software mechanism to override the TDR	write once, write 0 to 1 only
OTPEN[nb] (OTP Enable)	Stores information of which flash blocks is to be assigned as OTP (One Time Programmable) - nb means one register for each Lock register of the flash memory	write 0 to 1 only
TDR[n]_LOCK[m] Region Assignment	Stores information of which flash block is assigned to a TDR. • n specifies a TDR • m specifies the flash blocks as per the LOCK registers of the flash memory.	write once, write 0 to 1 only

79.5.1 Diary Base Address (DBA) DCF client

This DCF client holds base address information of the diary. The following diagram depicts this client.

CAUTION

This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.

DCF Client Address: 0x0000 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	DBA																															
Default	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 79-3. Diary Base Address (DBA) DCF client field description

Field	Description
0–31 DBA	Diary Base Address. This field holds the base address of the diary. This information is then used to verify if the diary being programmed matches any TDR region. **NOTE:** Diary Base Address must be on a 4 KB boundary.

79.5.2 Tamper Region Override (TO) DCF client

This DCF client holds override information for each TDR. The following figure shows the diagram for this register.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the override of a TDR permanent.

DCF Client Address: 0x0001 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 79-4. Tamper Region Override (TO) DCF client field descriptions

Field	Description
0–25 Reserved	This field is reserved and should be written with all zeroes when programming the TOE fields.
26–31 TOE5 to TOE0	TDR Override Enable. Each TOE*n* field, where *n* represents the TDRs, holds information of the Override Enable status for each TDR. 0 Normal operation. Blocks assigned to this TDR can NOT be erased until the diary is written. 1 Diary override. The blocks assigned to this TDR can be erased WITHOUT writing to the diary.

79.5.3 Software Tamper Region Override Disable (STO_DIS) DCF client

This DCF client holds information to disable the software mechanism to override the tamper detect regions.

CAUTION

This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.

DCF Client Address: 0x0002 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	STO_DIS5	STO_DIS4	STO_DIS3	STO_DIS2	STO_DIS1	STO_DIS0
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 79-5. Software Tamper Region Override Disable (STO_DIS) DCF client field descriptions

Field	Description
0–25 Reserved	This field is reserved and should be written with all zeroes when programming the STO_DIS fields.
26–31 STO_DIS5 to STO_DIS0	Software TDR Override Disable TDR*n*. Each STO_DIS*n* field, where *n* represents the TDR number, holds information to disable the software mechanism to override TDR*n*. 0 Software tamper detect override mechanism supported 1 Software tamper detect override mechanism disabled. Any attempt to override TDR*n* by loading a service key value in STO_KEY*n* register is ignored.

79.5.4 One Time Programmable Enable (OTPEN0) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK0 register of the flash memory. The following diagram depicts this client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x0008 Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	LOWOTPEN[11:0]												MIDOTPEN[15:0]															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 79-6. One Time Programmable Enable (OTPEN0) DCF client

Field	Description
0–3 Reserved	This field is reserved and should be written with all zeroes when writing to this client.
4–15 LOWOTPEN	Low Block OTP Enable. This field affects flash memory blocks in Low address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP
16–31 MIDOTPEN	Mid Block OTP Enable. This field affects flash memory blocks in Mid address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

79.5.5 One Time Programmable Enable (OTPEN1) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK1 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x0009 Access: DCF write

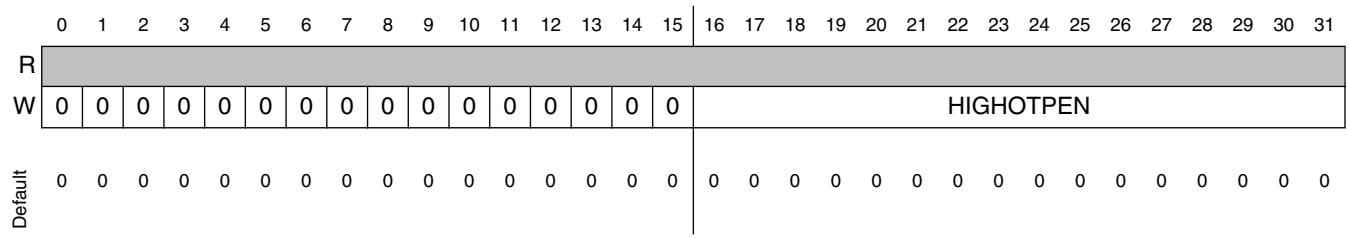


Table 79-7. One Time Programmable Enable (OTPEN1) DCF client field descriptions

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 HIGHOTPEN	High Block OTP Enable. This field affects flash memory blocks in High address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

79.5.6 One Time Programmable Enable (OTPEN2) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK2 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x000A Access: DCF write

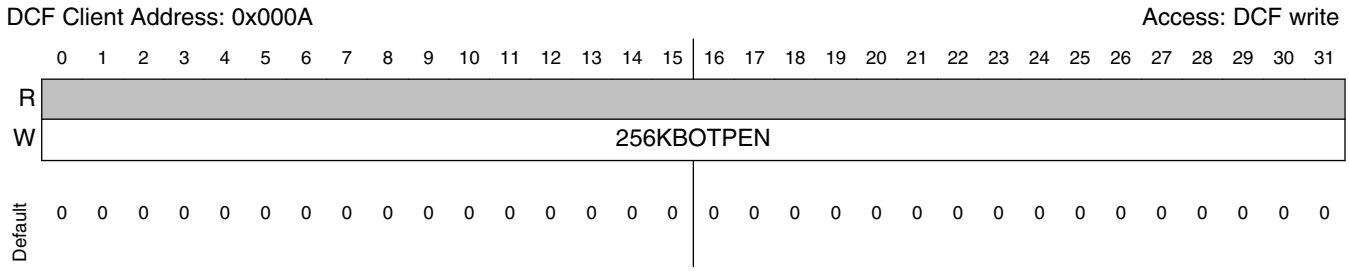


Table 79-8. One Time Programmable Enable (OTPEN2) DCF client field descriptions

Field	Description
0–31 256KBOTPEN	256 KB Block OTP Enable. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

79.5.7 One Time Programmable Enable (OTPEN3) DCF client

This DCF client holds OTP information for particular blocks of flash memory. This DCF client mirrors the LOCK3 register of the flash memory. The following diagram depicts the client.

CAUTION

This DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored, making the OTP assignment of a flash memory block irreversible.

DCF Client Address: 0x000B — Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	256KBOTPEN															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 79-9. One Time Programmable Enable (OTPEN3) DCF client field descriptions

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 256KBOTPEN	256 KB Block OTP Enable. This field affects flash memory blocks in the upper range of 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned as OTP 1 Corresponding flash blocks are assigned as OTP

79.5.8 Tamper Region Assignment DCF client (TDRn_LOCK0)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK0 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK0 register in the flash memory. The following diagram depicts the TDRn_LOCK0 DCF client.

CAUTION

This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.

DCF Client Address: 0x0014 + ((n x 4) + 0) Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	LOWTDRAGN												MIDTDRAGN															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 79-10. Tamper Region Assignment DCF client (TDRn_LOCK0)

Field	Description
0–3 Reserved	This field is reserved and should be written with zeroes when writing to this client.
4–15 LOWTDRAGN	Low Block Tamper Region Assignment. This field affects flash memory blocks in Low address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn
16–31 MIDTDRAGN	Mid Block Tamper Region Assignment. This field affects flash memory blocks in Mid address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

79.5.9 Tamper Region Assignment DCF client (TDRn_LOCK1)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the

LOCK1 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK1 register in the flash memory. The following diagram depicts the TDRn_LOCK1 DCF client.

CAUTION

This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.

DCF Client Address: 0x0014 + ((n x 4) + 1) Access: DCF write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R																																
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	HIGHTDRAGN															
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 79-11. Tamper Region Assignment DCF client (TDRn _LOCK1)

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 HIGHTDRAGN	High Block Tamper Region Assignment. This field affects flash memory blocks in High address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

79.5.10 Tamper Region Assignment DCF client (TDRn_LOCK2)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK2 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK2 register in the flash memory. The following diagram depicts the TDRn_LOCK2 DCF client.

CAUTION

This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.

DCF Client Address: 0x0014 + ((n x 4) + 2) Access: DCF write

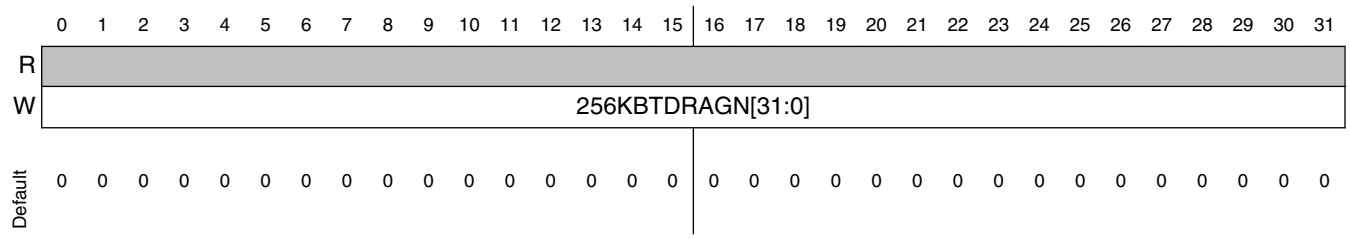


Table 79-12. Tamper Region Assignment DCF client (TDRn_LOCK2)

Field	Description
0–31 256KBTDRAGN[31:0]	256 KB Block Tamper Region Assignment. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

79.5.11 Tamper Region Assignment DCF client (TDRn_LOCK3)

This DCF client holds the assignment information of flash memory blocks to a specific TDR. Each flash memory block can be assigned to any TDR (*n* varies from 0 to 5). This DCF client holds the assignment information for only those blocks specified in the LOCK3 register in the flash memory. For that reason, the layout of this DCF client mirrors the LOCK3 register in the flash memory. The following diagram depicts the TDRn_LOCK3 DCF client.

CAUTION

This write-once DCF client can be written only one time. After one DCF record has written to this client, all subsequent writes are ignored.

DCF Client Address: 0x0014 + ((n x 4) + 3) Access: DCF write

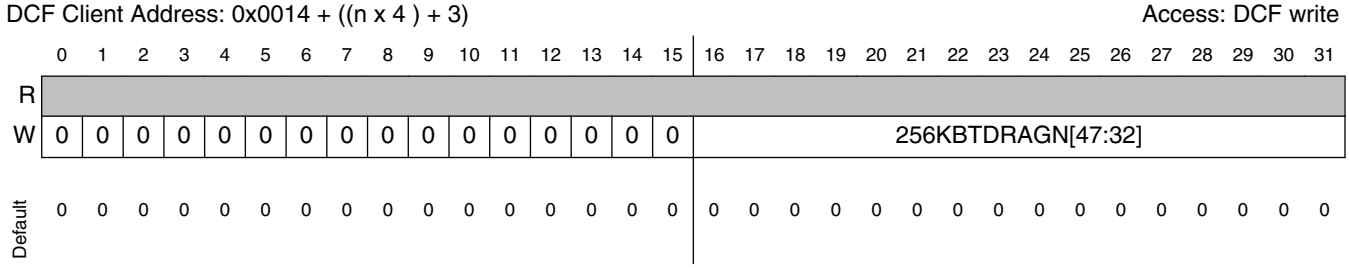


Table 79-13. Tamper Region Assignment DCF client (TDRn_LOCK3)

Field	Description
0–15 Reserved	This field is reserved and should be written with zeroes when writing to this client.
16–31 256KBTDRAGN[47:32]	256KB Block Tamper Region Assignment. This field affects flash memory blocks in 256KB address space. See the chip-specific information for the mapping between field bits and flash blocks. The same mapping also applies to control similar functions in the flash memory module. 0 Corresponding flash blocks are not assigned to the TDRn 1 Corresponding flash blocks are assigned to the TDRn

79.6 Memory Map/Register Definition

TDM memory map

Address offset (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	TDR Status Register (TDM_TDRSR)	32	R	See section	79.6.1/3347
4	Last Flash Programmed Address Register (TDM_LFPAR)	32	R	0000_0000h	79.6.2/3350
8	Diary Base Address (TDM_DBA)	32	R	See section	79.6.3/3350
10	Software Tamper Override Key Region (TDM_STO_KEY0)	32	R/W	0000_0000h	79.6.4/3351
14	Software Tamper Override Key Region (TDM_STO_KEY1)	32	R/W	0000_0000h	79.6.4/3351
18	Software Tamper Override Key Region (TDM_STO_KEY2)	32	R/W	0000_0000h	79.6.4/3351
1C	Software Tamper Override Key Region (TDM_STO_KEY3)	32	R/W	0000_0000h	79.6.4/3351
20	Software Tamper Override Key Region (TDM_STO_KEY4)	32	R/W	0000_0000h	79.6.4/3351
24	Software Tamper Override Key Region (TDM_STO_KEY5)	32	R/W	0000_0000h	79.6.4/3351

79.6.1 TDR Status Register (TDM_TDRSR)

This register contains the current status, i.e., locked or unlocked, of each tamper region, i.e., TDR.

- If a tamper region is locked, the blocks assigned to that tamper region cannot be erased.
- If a region is unlocked, the blocks within the tamper region can be erased.

The register contains a status bit for each TDR. Status is determined as follows:

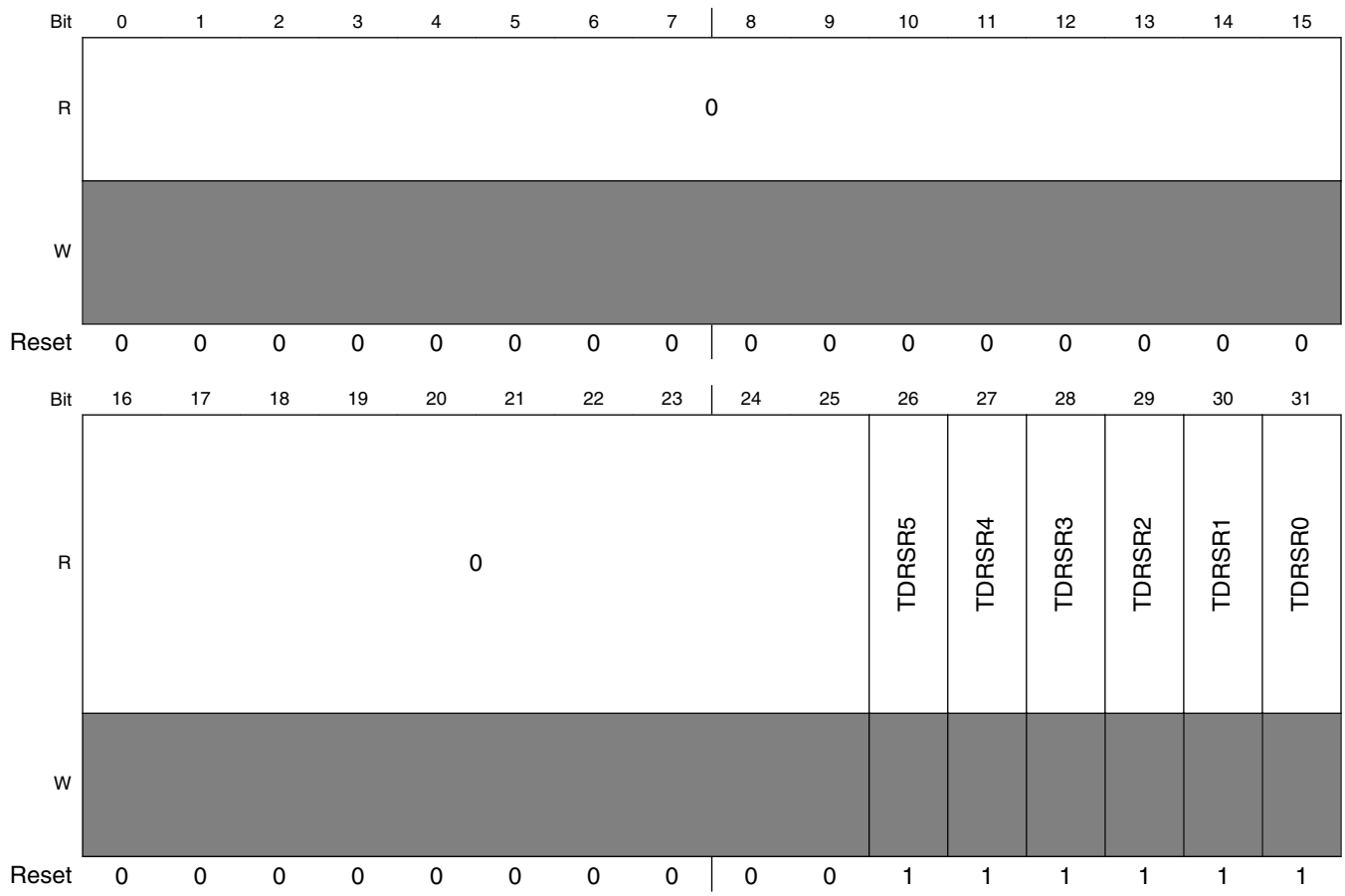
1. The default status for each TDR is ‘1’, which indicates that the erase operation is disabled for all flash blocks associated with the TDR.

2. The status bit for a TDR is set to ‘0’ (indicating that the erase operation for all flash blocks associated with the TDR is enabled) if either a successful programming operation to the TDR diary region has been performed or the TDR Override bit for that TDR is set.

NOTE

- TDRSR is a read-only register and writes have no effect on it. Write operations result in a transfer error.
- If the reset value for this register is determined by a DCF record, it will be noted in the chip-specific information at the beginning of this chapter. Otherwise, the reset value is as shown.

Address: 0h base + 0h offset = 0h



TDM_TDRSR field descriptions

Field	Description
0–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 TDRSR5	Represents the status of TDR 5 in conjunction with the Diary Override Enable status.

Table continues on the next page...

TDM_TDRSR field descriptions (continued)

Field	Description
	0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
27 TDRSR4	Represents the status of TDR 4 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
28 TDRSR3	Represents the status of TDR 3 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
29 TDRSR2	Represents the status of TDR 2 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
30 TDRSR1	Represents the status of TDR 1 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.
31 TDRSR0	Represents the status of TDR 0 in conjunction with the Diary Override Enable status. 0 TDR is unlocked. Blocks assigned to this TDR are NOT blocked for erasing by the TDM module. This unlock state is achieved by a successful programming operation to the correct diary region or this TDR has been permanently unlocked by a DCF record that writes to the Tamper Region Override DCF register or a successful software override operation. 1 TDR is locked. Erase of blocks assigned to this TDR is not possible.

79.6.2 Last Flash Programmed Address Register (TDM_LFPAR)

This register holds information of the address of the last successful programming operation provided from the flash memory to the TDM. Writes to this register are ignored and will generate a transfer error.

Address: 0h base + 4h offset = 4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								LFPAR																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM_LFPAR field descriptions

Field	Description
0–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
8–31 LFPAR	Last Flash Programmed Address. These bits represent the address of the last successful programming operation.

79.6.3 Diary Base Address (TDM_DBA)

This register holds base address information of the diary. Any write to this register is ignored and generates a transfer error.

Address: 0h base + 8h offset = 8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DBA																															
W																																
Reset	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*	x*

* Notes:
- The reset value of this register is set via DCF record.x = Undefined at reset.

TDM_DBA field descriptions

Field	Description
0–31 DBA	Diary Base Address. These bits represent the address of the diary location in the flash. **NOTE:** The diary base address is established via the DBA DCF record. The reset value of this register is determined by that record.

79.6.4 Software Tamper Override Key Region (TDM_STO_KEY*n*)

The STO_KEYn register holds the service key for overriding the Tamper Detect Region n, where n denotes the Tamper Detect Region number.

Caution

1. The STO_KEYn registers can only be written once.
2. The associated DCF client is writable from 0 to 1 only. Attempted writes of bits from 1 to 0 are ignored.

Address: 0h base + 10h offset + (4d × i), where i=0d to 5d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	STO_KEY																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TDM_STO_KEY*n* field descriptions

Field	Description
0–31 STO_KEY	Software Tamper Override Key Diary n. This field holds the service key value for overriding Tamper Detect Region n. By loading STO_KEYn with the correct signature of 0x55AA5A5A, Tamper Detect Region n is overridden, and the flash blocks assigned to Tamper Detect Region n are open for erase. This field has no effect when STO_DISn field of STO_DIS DCF client is set. **Caution:** The STO_KEYn registers can only be written once.

79.7 Functional description

The primary purpose of the TDM is to provide a Tamper Detection mechanism by enabling an Erase Counter (or diary) for regions within the flash. These regions are called Tamper Detection Regions. The following sections cover the entire functionality of this module.

79.7.1 Flash erase counter and tamper detection

A flash region block or part of it is assigned for implementing the diary. The base address for the diary is defined by a DCF Record.

The diary is divided into 6 parts where each part consists of 256 × 64-bit records, to record up to 256 write attempts on the flash blocks assigned to each Tamper Detection Region. For correct operation, the diary should reside in a flash block that is assigned as OTP.

Each flash block can be assigned to any of the TDRs. Related to each TDR is a group of 4 × 32-bit internal hardware registers that are initialized by DCF records that define the tamper region. These registers mirror exactly the LOCK registers bits from the flash module.

Each TDR*n*_LOCK*m* register is loaded at boot time by the SSCM, which defines if the associated flash block is Locked for Erase in a particular TDR:

- ’1’ means the flash block is locked for erase
- ’0’ means the flash block is not locked for erase

Before performing an erase in one of the blocks locked for erase and associated to a TDR, a program operation has to be performed on the diary part for that TDR. As the diary should be OTP, successive writes are allowed on unprogrammed double words only. This ensures that diary records can only increase and that data recorded in the diary is not modifiable by software.

At the end of a successful double-word program operation, the flash module indicates to the TDM the address of the programmed double word. The TDM determines whether the programmed data was written to the correct TDR diary location and then unlocks the blocks for erase operation for the corresponding TDR. The TDM ensures that this erase operation on unlocked blocks is allowed until the completion of the erase sequence or until next desctructive reset.

When attempting to update the TDM diary and erase flash blocks, set MCR[PECIE] in the Embedded Flash Memory registers to enable interrupt notification for program and erase operations. Then clear MCR[PECIE] to service the interrupt after each program or erase operation. This sequence must be executed before attempting to update the TDM diary, and then again before attempting to erase the unlocked flash blocks.

NOTE

All blocks associated to a TDR where an erase operation is allowed can be erased all together.

The data programmed in the diary location is flexible, and may include count information or other important customer information.

NOTE

There is no restriction on the type or format of data that can be programmed in the diary, so software must verify whether the diary is full. If a TDR's diary area is full, the tamper detect mechanism can be disabled for that TDR by creating a Tamper Region Override (TO) DCF record with the TDR's corresponding TDR Override Enable (TOE) bit set to '1'. This

allows erase operations to be performed to the TDR without any diary entry.

The setting from “override tamper region lock” DCF record (Diary) has priority over the settings in the TDRn_LOCK0 – 3 registers. In other words, if a TDR is unlocked for erase by the override tamper region lock, the TDR*n*_LOCK*m* bits have no effect on the TDR that has been overridden, but the tamper detection remains active for all TDRs not having the override bit set.

Moreover, if a block is locked by the PASS_LOCK*x*_PG*n* registers or by TDR*n*_LOCK*m*, the block is locked for erase.

NOTE

The settings in the Tamper Region Override Register have priority over the TDR*n*_LOCK*m* registers. The PASS module also has priority over the settings done in the TDM. For instance, if blocks in a TDR are unlocked for erase operation via the TDM, but the same blocks are programmed to be locked for erase operation in PASS, the blocks end up being locked for erase operation.

Attempts to erase the flash without first writing the diary will be ignored by the flash. The contents in the flash block will remain unchanged and MCR[PEG] will be asserted, indicating the operation was successful and the contents of the block are properly protected from the erase operation.

79.7.2 Assigning blocks to Tamper Detect Regions (TDR)

Each TDR is defined by a group of 4 x 32-bit DCF records and clients, with each containing bits that map to specific blocks in a flash region. These mappings mirror exactly the LOCK registers bits from the flash module. Hence, the block assignment registers are named as TDR*n*_LOCK*m* where *n* is the number of a tamper region, that is, 0- 5 and *m* varies from 0-3 as flash contains four Lock registers.

In these DCF records:

- ’1’ means that the corresponding flash block is assigned to this TDR
- ’0’ means that the corresponding flash block is not assigned to this TDR (Default)

DCF clients are not visible to software. They can not be read nor written. They can only be initialized by DCF record within the UTest flash block during reset. The value of these registers can be evaluated by reading all the UTest DCF records and searching for records that correspond to these clients.

NOTE

No flash block should be assigned to more than one TDR. Each block should correspond to a single TDR. For example, if a single block is assigned to TDR0 and TDR1, then even if the Diary is programmed for both TDR0 and TDR1, this flash block will never be unlocked for erase operation.

79.7.3 Diary

The diary is a region of the flash memory where records of block erases for each TDR are stored. The format and size of the records are not fixed. The only requirements are that each diary has a maximum size of 2 KB, and the minimum size of any programming operation is 8 bytes to honor OTP restrictions. Therefore, each diary can hold a maximum of 256 64-bit entries.

A flash region block or part of it must be assigned for implementing the diary. The base address for the diary is determined by a value written in a DCF record. The diary can be placed within any flash block in flash array, and history records in it can be read by any core and from the HSM.

NOTE

To maintain the security of the TDR, the block where the diary is placed must be assigned as OTP within the OTP registers.

The base address of the diary is implemented using a DCF Client. The DCF client is only writable once and the default bit state is 1.

The DCF clients can not be read nor written by software. They can only be initialized by DCF record from within UTEST block during reset. The value of these registers can be evaluated by reading all the UTEST DCF records and searching for records that write these DCF clients.

79.7.4 Specifying the diary base address

The diary is a region of flash memory where records of block erases for each TDR are stored. The format and size of these records is defined by the customer. The only hardware constraints are that each diary has a maximum size of 2 KB and the minimum size of any programming operation is 8 bytes. Therefore, each diary region limits the erases of the assigned blocks to 256 erases.

The diary can be placed within any flash block in the flash array. To maintain the security of the TDR, the block where the diary is placed, must be assigned as OTP within the OTP registers, while in customer delivery or OEM production life cycle states.

The base address of the diary is implemented with a 20-bit DCF client that can be initialized by the SSCM during reset. The DCF client is writable once and the default bit state is ‘1’.

The base address register is not visible to software. It cannot be read nor written. It can only be initialized by DCF record within UTest flash during reset. The value of this register can be evaluated by reading all the UTest flash DCF Records and searching for record that writes this register.

Each TDR is assigned a 2 KB section of the diary. With 6 TDRs, 12 KB of diary space is required.

The diary base address must be at a 4 KB boundary. Therefore the least significant 12 bits of the base address must all be ‘0’.

The Base address of the diary is represented by:

Diary_Base_Address = 0bxxxx_xxxx_xxxx_xxxx_xxxx_0000_0000_0000

The base address of each TDR is shown below:

- TDR0 = Diary_Base_Address + 0b00_0000_0000_0000
- TDR1 = Diary_Base_Address + 0b00_1000_0000_0000
- TDR2 = Diary_Base_Address + 0b01_0000_0000_0000
- TDR3 = Diary_Base_Address + 0b01_1000_0000_0000
- TDR4 = Diary_Base_Address + 0b10_0000_0000_0000
- TDR5 = Diary_Base_Address + 0b10_1000_0000_0000
- End of the diary = Diary_Base_Address + 0b11_0000_0000_0000

79.7.5 TDR lock status

The status of whether a TDR is locked for erase is maintained by the TDM. This is accomplished by implementing a register with as many bits as the number of TDRs. The default state of these bits is 1, which is defined as "TDR Erase Blocked". See TDR Status Register (TDM_TDRSR) for further details about this register.

When the flash memory module has completed a programming operation, the address of the location programmed is output from the flash module. The TDM in turn compares that address to determine if it falls within the range of any TDR diary regions. If it is within any TDR, the corresponding status bit is set to 0 meaning "TDR Erase Enabled".

These status bits are all reset to 1 when the flash module signals an erase complete by asserting the PEC (Program Erase Complete) signal.

The state of the 6 status bits can be read through the TDRSR. See TDR Status Register (TDM_TDRSR). The address of the last successful programming operation provided from the flash memory to the TDM can be read through the LFPAR. See Last Flash Programmed Address Register (TDM_LFPAR).

79.7.6 TDR override

Before a block protected by a tamper detect region can be erased, a record must be written to the associated diary. Software must search through the diary to find the end of the records, where the next available diary location that can be written. After a maximum of 256 records are programmed to the diary, a TDR diary is full. Full means that at least 1 bit has been programmed to ‘0’ in each 64 bit double word within the 2 KB diary.

If software determines that a particular TDR diary is full it may override the diary to allow erasing to continue or it may choose to stop further erases and to flag an error.

Diary Override bits are implemented with a DCF client consisting of 6 bits, one for each TDR. See the device configuration information at the beginning of this chapter for TDM DCF client details.

The Diary Override bits are by default set to ‘0’; which means the diary must be written before an erase can occur. The DCF client can be written many times but the bits can only be written to ‘1’. Writes to ‘0’ are ignored. In this way, by default, all Tamper Detect Regions are active. The Tamper detect region diaries can be individually overridden by writing a DCF record that sets that corresponding bit to ‘1’. In this mode, the TDR status bit is permanently set to 0 to allow flash erase.

The application can also override the diary by loading a service key to the STO_KEY*n* program-visible register, where n denotes the tamper detect region. By loading the value 0x55AA5A5A into STO_KEYn register, the corresponding tamper detect region is overridden. This software override mechanism is disabled if the STO_DISn field of the STO_DIS DCF Client is set.

79.7.7 One time programmable (OTP)

79.7.7.1 Overview

Any flash block within the flash array can be assigned, at any time, to be OTP. Once a flash block is assigned OTP, it cannot be changed back.

OTP means that flash erase of the entire block is disabled and only 64-bit double words that are already erased (that is, flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block will become OTP after the next functional reset.

Chapter 80
Flash Memory Programming and Configuration

80.1 Introduction

This chapter provides basic guidance on activities related to configuring and programming the embedded flash memory in the MPC5746C microcontroller.

Multiple layers of protection are available to prevent unintended updates to flash contents and unauthorized reads and writes to flash contents. The protection mechanisms range from simple write-protect bits for individual flash blocks to four levels of 256-bit password read and/or write protection and configuration of flash memory blocks as One Time Programmable (OTP).

Additionally, the MPC5746C microcontroller includes a tamper detection feature, debug port enable/disable, and the ability to disable the ability for the device to be put into factory test mode.

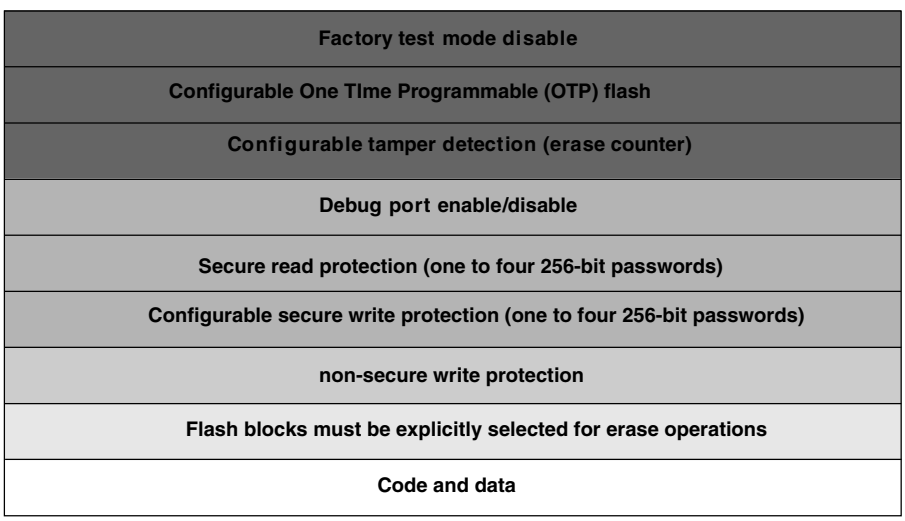


Figure 80-1. Flash memory protection layers

At the lowest layer of protection, all flash erase operations require the explicit selection of one or more blocks of flash memory before the operation is performed. When executed, the operation is only performed on the selected block(s).

The next layer is the chip's ability to lock individual flash blocks against unintended programming and erase operations. This level of protection does not offer any password protection and is intended to provide protection against *accidental* changes to code and data.

The first layer of *secure* protection is the microcontroller's ability to support up to four levels of 256-bit password-secured *write* protection for individual flash blocks.

Similarly, up to four levels of password-secured *read* protection can be implemented. Unlike the password-secured write protection, which is implemented on a per-block basis, password-secured read protection applies to five pre-defined (at the factory) groups of flash blocks. The read protection applied to a group applies to each block within the group. In contrast, secure write protection is defined at the individual block level.

Secure password protection can also be implemented for the debug port.

The MPC5746C microcontroller also includes a tamper detection feature. In its most basic implementation, tamper detection acts as an erase counter. If tamper detection is implemented, a DCF record in the tamper detection *diary* must be written by software before a protected flash block can be erased. Multiple tamper detection regions can be defined by the customer, with each region containing one or more blocks. A block can be included in more than one tamper detection region if desired.

Another protection feature related to tamper detection is the ability to configure any flash block as one-time programmable (OTP).

Finally, the microcontroller includes a feature that allows the customer to disable the ability to place the device in factory test mode.

The remainder of the chapter provides details regarding these protection mechanisms and gives a brief overview of security configuration planning.

Note

Most of the topics in this chapter build on information presented in previous topics. To get a clear overview of how the flash protection features work and relate to each other, please read the chapter in sequence from beginning to end.

Access protection configuration is applied on a per-block basis (except read protection and tamper detection applied to groups of flash memory blocks), i.e., a custom level of access protection can be provided for each flash memory block. This is accomplished

through the use of registers containing bits that are mapped to individual flash blocks. Understanding the mapping is key to understanding the implementation and use of the MPC5746C microcontroller's flash protection features.

Depending on the type of protection to be applied, different sets of registers (or even DCF records) are used, but the mapping scheme is consistent. This mapping scheme is first presented in Selection of flash memory blocks for erase. It is important to understand the topic before proceeding to other sections.

80.2 Selection of flash memory blocks for erase

The registers used for selecting flash blocks for erase operations are the flash control registers named LOCK0, LOCK1, and LOCK2. There is also a LOCK3 register, but no flash blocks are mapped to it.

The LOCK0–2 registers are 32-bit registers containing fields associated with specific address spaces: low, mid, high, and 256 KB. Each non-reserved bit in the fields is mapped to a specific flash block.

For example, Figure 80-2 shows the Select 0 (SEL0) register for the flash module. This register provides a way of selecting blocks in the low and mid address spaces for erase operations.

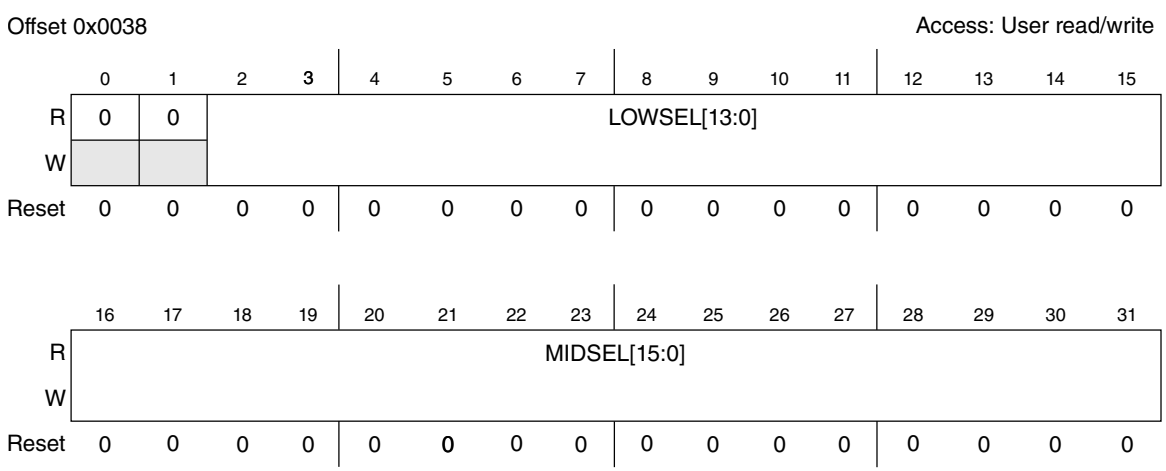


Figure 80-2. SEL0 flash control register

Table 80-1 shows an example of a mapping of SEL0 register's LOWSEL and MIDSEL field bits to individual flash blocks[1]. In this case, not all bits of the LOWSEL and MIDSEL fields are mapped—only LOWSEL bits 0–9 (register bits 15–6,[2]) and

1. The table lists the LOWLOCK register field but the structure is identical to the LOWSEL field in Figure 80-2.
2. Register bits in Power Architecture-based microcontrollers are numbered with the most significant bit as 0. This is not necessarily the case with register field bit numbering. The field bit numbering shown in the figure has the least significant bit numbered as 0.

MIDSEL bits 0–7 are mapped. Before any mapped block can be erased, it must be selected by writing a '1' to its corresponding bit in one of the SEL*n* registers. See the Embedded Flash Memory configuration for the complete mapping.

Note

Values written to the SEL registers do not persist through device reset. As indicated by the reset values shown in Figure 80-2, all bits are reset to '0'. As a result, after a device reset all blocks mapped to the register are, by default, not selected for erase.

Table 80-1. Flash address space and flash lock bit mapping

Start Address	End Address	Size	Partition	LOCKn Register[1]	LOCKn Register Field bit	LOCKn DCF Record DATA bit[2]	Read Lock Group[3]	Test Mode Disable Block Select Bit	Description
0x00400000	0x00403FFF	16	-	LOCK 0	TSLOCK	31	0	-	UTest NVM Block Space[4]
0x00620000	0x0062FFFF	64	1	LOCK 0	LOWLOCK[9]	25	3	14	64KB HSM code block 3
0x00FF0000	0x00FFFFFF	64	1	LOCK 0	LOWLOCK[8]	24	1	15	64KB code block
0x00610000	0x0061FFFF	64	0	LOCK 0	LOWLOCK[7]	23	3	0	64KB HSM code block 2
0x00FE0000	0x00FEFFFF	64	0	LOCK 0	LOWLOCK[6]	22	1	1	64KB code block
0x00FD8000	0x00FDFFFF	32	1	LOCK 0	LOWLOCK[5]	21	1	2	32KB Code Flash block
0x00FD0000	0x00FD7FFF	32	1	LOCK 0	LOWLOCK[4]	20	1	3	32KB Code Flash block
0x00FC8000	0x00FCFFFF	32	0	LOCK 0	LOWLOCK[3]	19	1	4	32KB Code Flash block
0x00FC0000	0x00FC7FFF	32	0	LOCK 0	LOWLOCK[2]	18	1	5	32KB Code Flash block
0x00404000	0x00407FFF	16	1	LOCK 0	LOWLOCK[1]	17	-	6	16KB BAF block[5]
0x00F8C000	0x00F8FFFF	16	0	LOCK 0	LOWLOCK[0]	16	3	7	16 KB Code Flash Block
0x00FAC000	0x00FAFFFF	16	3	LOCK 0	MIDLOCK[7]	7	1	16	16KB Code Flash block
0x00FA8000	0x00FABFFF	16	3	LOCK 0	MIDLOCK[6]	6	1	17	16KB Code Flash block
0x00FA4000	0x00FA7FFF	16	3	LOCK 0	MIDLOCK[5]	5	1	18	16KB Code Flash block
0x00FA0000	0x00FA3FFF	16	3	LOCK 0	MIDLOCK[4]	4	1	19	16KB Code Flash block

Table continues on the next page...

Table 80-1. Flash address space and flash lock bit mapping (continued)

0x00F9C000	0x00F9FFFF	16	2	LOCK 0	MIDLOCK[3]	3	1	20	16KB Code Flash block
0x00F98000	0x00F9BFFF	16	2	LOCK 0	MIDLOCK[2[	2	1	21	16KB Code Flash block
0x00F94000	0x00F97FFF	16	2	LOCK 0	MIDLOCK[1]	1	1	22	16KB Code Flash block
0x00F90000	0x00F93FFF	16	2	LOCK 0	MIDLOCK[0]	0	1	23	16KB Code Flash block
0x00F84000	0x00F87FFF	16	5	LOCK1	HIGHLOCK[1]	1	4	38	16KB HSM Data Block 1
0x00F80000	0x00F83FFF	16	4	LOCK1	HIGHLOCK[0]	0	4	39	16KB HSM Data Block 0
0x01240000	0x0127FFFF	256	7	LOCK2	256KLOCK[9]	9	1	62	256KB Code Flash block9
0x01200000	0x0123FFFF	256	7	LOCK2	256KLOCK[8]	8	1	63	256KB Code Flash block8
0x011C0000	0x011FFFFF	256	6	LOCK2	256KLOCK[7]	7	1	48	256KB Code Flash block7
0x01180000	0x011BFFFF	256	6	LOCK2	256KLOCK[6]	6	1	49	256KB Code Flash block6
0x01140000	0x0117FFFF	256	6	LOCK2	256KLOCK[5]	5	1	50	256KB Code Flash block5
0x01100000	0x0113FFFF	256	6	LOCK2	256KLOCK[4]	4	1	51	256KB Code Flash block4
0x010C0000	0x010FFFFF	256	6	LOCK2	256KLOCK[3]	3	1	52	256KB Code Flash block3
0x01080000	0x010BFFFF	256	6	LOCK2	256KLOCK[2]	2	1	53	256KB Code Flash block2
0x01040000	0x0107FFFF	256	6	LOCK2	256KLOCK[1]	1	1	54	256KB Code Flash block1
0x01000000	0x0103FFFF	256	6	LOCK2	256KLOCK[0]	0	1	55	256KB Code Flash block0

1. There are multiple sets of LOCKn registers, i.e., the flash module's LOCK0–-3 registers and PASS module's PASS_LOCK0–3_PG0–3, but all use the same mapping of register bits to flash blocks. The same mapping applies the the flash module's SEL0–3 registers.
2. The DCF record DATA bit refers to the DATA[31:0] field contained in DCF records created during OEM configuration activities. The DCF records related to flash blocks are used to implement password-secured write protection, passwordsecured read protection, tamper detection, and debug port enable/disable as a permanent part of the device configuration.
3. The Read Lock group pertains to protecting code and data blocks from being read by debuggers. This type of protection is implemented using DCF records and the PASS_LOCK3_PG0–3 registers.
4. One time programmable.
5. The Boot Assist Flash (BAF) is one time programmable and the majority of it is programmed in the factory. The BAF handles the initial start up of the device.

Example: 80.2.1 Selecting blocks for erase

Selecting blocks for erase

Referring to Table 80-1, writing the value 0x0030_0000 to the SEL0 register (Figure 80-2) selects the following blocks for erase:

Start Address	End Address	Size (KB)	Partition	LOCK*n* Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit	Read Lock group	Test Mode Disable Block Select Bit	Description
0x00FD0000	0x00FD7FFF	32	1	LOCK0	LOWLOCK[4]	20	1	3	32KB Code Flash block
0x00FD8000	0x00FDFFFF	32	1	LOCK0	LOWLOCK[5]	21	1	2	32KB Code Flash block

80.3 Non-secure write protection

Non-secure write protection refers to the chip's ability to lock individual flash blocks against programming and erase operations. This level of protection is intended to provide protection against *accidental* changes to code and data. Although similar in function, this is *not* associated with the built-in register protection (REG_PROT) feature included with many of the chip's modules. The non-secure write protection feature detailed here applies only to embedded flash memory.

Implementing non-secure write protection is almost identical to selecting specific flash blocks for an erase operation—the only difference is the register used. To provide non-secure write protection for a block, write a '1' to the corresponding mapped bit found in one of the LOCK0–2 flash control registers.

Note

As indicated by the reset values shown in Figure 80-3, after reset all blocks mapped to the register are, by default locked against programming and erase. Values written to the LOCK and SEL registers do not persist through device reset.

The same bit mapping that applies to the SEL0–3 registers also applies to the LOCK0–3 registers.

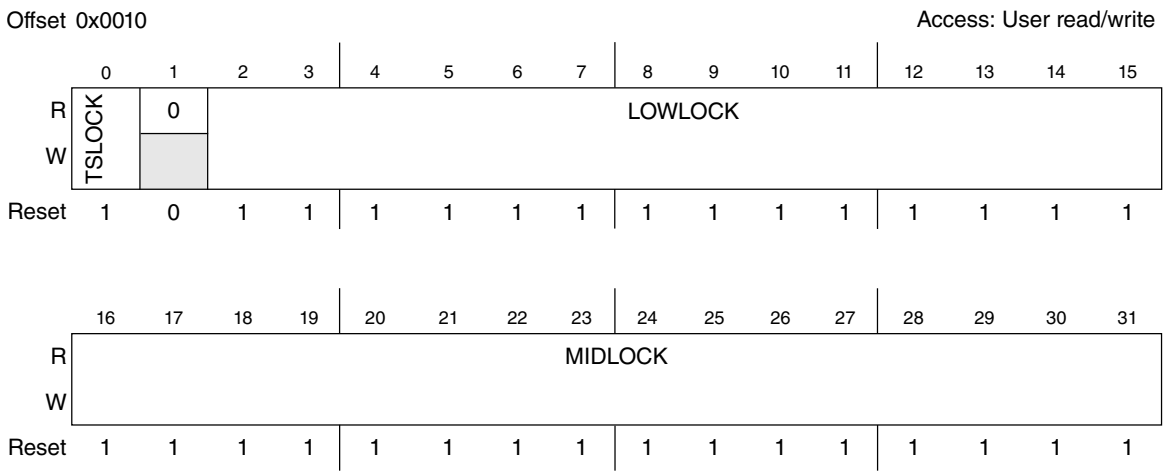


Figure 80-3. LOCK0 flash control register

Example: 80.3.1 Selecting

Selecting

Referring to Table 80-2, writing the value 0x0030_0000 (same value used in Selecting blocks for erase) to the LOCK0 register (Figure 80-3) locks the following blocks (same blocks selected in Selecting blocks for erase) so they cannot be programmed or erased, even if selected for erase:

Start Address	End Address	Size (KB)	Partition	LOCK*n* Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit	Read Lock group	Test Mode Disable Block Select Bit	Description
0x00FD0000	0x00FD7FFF	32	1	LOCK0	LOWLOCK[4]	20	1	3	32KB Code Flash block
0x00FD8000	0x00FDFFFF	32	1	LOCK0	LOWLOCK[5]	21	1	2	32KB Code Flash block

As long as the LOCK0[LOWLOCK] bits are unchanged, the above blocks cannot be modified unless the protection is overridden (discussed in Secure write protection). All other blocks mapped to bits in the LOCK0 *can* be programmed because their corresponding bits have been set to '0'.

80.4 Secure write protection

Secure write protection refers to the chip's ability to implement up to four levels of 256-bit password-secured *write* protection for individual flash blocks. Use of this feature is optional but the default behavior of the MPC5746C microcontroller is to apply secure write protection. If there is a need to *not* write-protect specific blocks, that needs to be configured by creating the appropriate DCF records.

Note that the *secure* write protection settings override the non-secure write protection detailed in Non-secure write protection.

Implementing a custom secure write protection scheme requires creating DCF records that define the level of write protection to be applied to each flash block. The DCF records use a bit mapping scheme identical to the mapping in the flash module's SEL*n* and LOCK*n* registers. The write protection scheme becomes a permanent part of the device but blocks can be temporarily unlocked. Depending on the specific implementation, temporarily unlocking the write protection requires matching up to four 256-bit passwords.

The easiest way to understand the secure write protection functionality is to examine the role played by the PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers.

The PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers are four identical sets of registers contained in the PASS module—they distinguished by the PG*n* suffix. The PG*n* (Password Group *n*) suffix indicates the password required to unlock that set of registers. Password 0 unlocks PASS_LOCK0_PG0, PASS_LOCK1_PG0, PASS_LOCK2_PG0, and PASS_LOCK3_PG0. Similarly, the registers suffixed with PG1 are unlocked by matching password 1, and so on.

The PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* register sets have two functions related to secure write protection:

- Reading the registers provides the secure write lock status for each flash block.
- Changing the value of a mapped bit in a register changes the secure write protect status of the corresponding flash block.

Each set of registers contains the same mapping of bits to flash blocks as the flash module's SEL*n* and LOCK*n* registers. In any of the four PASS_LOCK*x*_PG*n* register sets, if a mapped bit is set to '1', locking a flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access to the register, and setting the bit to '0'. If the same bit is set to '1' in more than one

PASS_LOCK*x*_PG*n* register set, each must be unlocked using the corresponding password and the bit must be set to '0' before the flash block is available for programming or erase.

Figure 80-4 shows how the registers interact. Note that secure write protection is only affected by the settings in PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n*. The PASS_LOCK3_PG*n* registers have additional security functions, discussed in later topics.

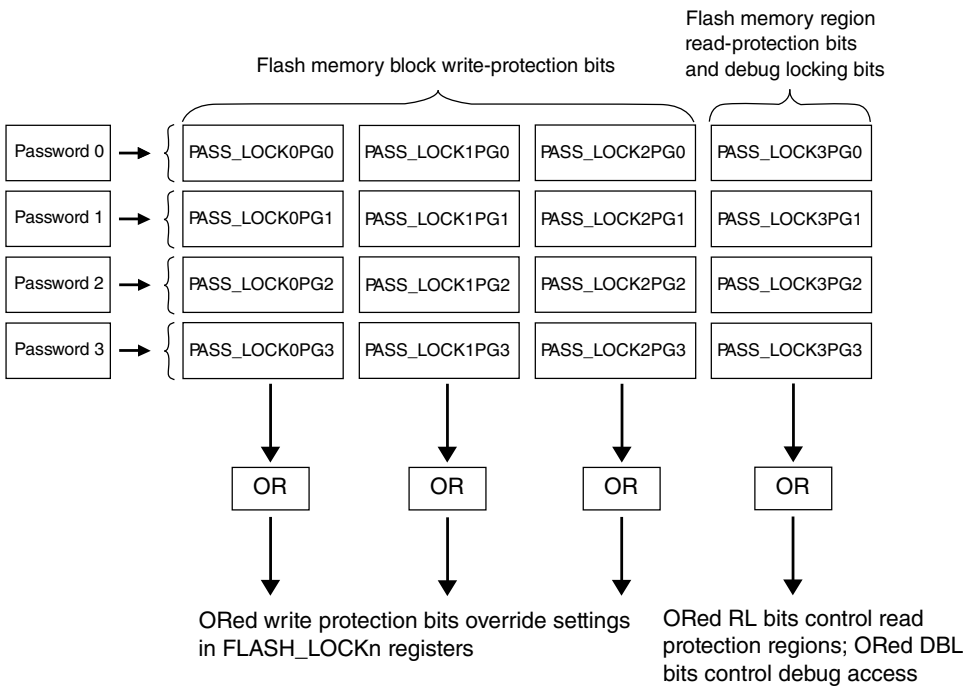


Figure 80-4. Secure write protection

The next sections discuss implementing secure write protection and overriding secure write protection.

80.4.1 Implementing secure write protection

Implementing secure write protection is a one-time task performed either at the factory or by the customer during chip configuration activities. It normally requires creating a DCF record for each of the PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n* registers[3]. The DCF records have a structure that includes the same mapping of bits to flash memory blocks as

3. The default behavior for secure write protection in the case where passwords are created but no secure write protection DCF records are created, is for all blocks to have four levels of password-secured write protection. To vary the protection among all blocks, a DCF record for each password group must be created.

the flash module's SEL*n* and LOCK*n* registers. The values assigned become the permanent register reset values for the PASS_LOCK0_PG*n*–PASS_LOCK2_PG*n* registers.

See Device Configuration Format (DCF) Records for complete details on DCF records. This section summarizes the contents of a DCF record as required to implement secure write protection.

Even though the mapping of bits to flash memory blocks is identical to the mapping used for selecting flash blocks for erase of non-secure write locking, there are two significant differences:

1. Configuring flash blocks for secure write protection is done only during device configuration to customer specifications, either at the factory or by the customer during OEM production activities. The selection becomes a permanent part of the device configuration.
2. Configuration of blocks for secure write protection is done by writing *DCF records*, which is why the selection becomes a permanent part of the device.

CAUTION

Careful planning is required before writing DCF records. They are created in a write-once area of flash. Multiple instances of any DCF record can be created, but in some cases only the record created first is used and with others only the most-recently written record is used. Be sure to understand the characteristics of a DCF record before creating it.

The DCF records required for implementing a secure write protection scheme are detailed in the PASS section of DCF client list table in the attached DCF_clients.xlsx. Table entries provide the CS and Address value combinations that correspond to PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers.

The format of a secure write protection record is summarized here for convenience.

DCF records appear as contiguous double-word (64-bit) entries programmed in a reserved area of OTP UTEST flash memory. The structure of a DCF Record is shown in Figure 80-5.

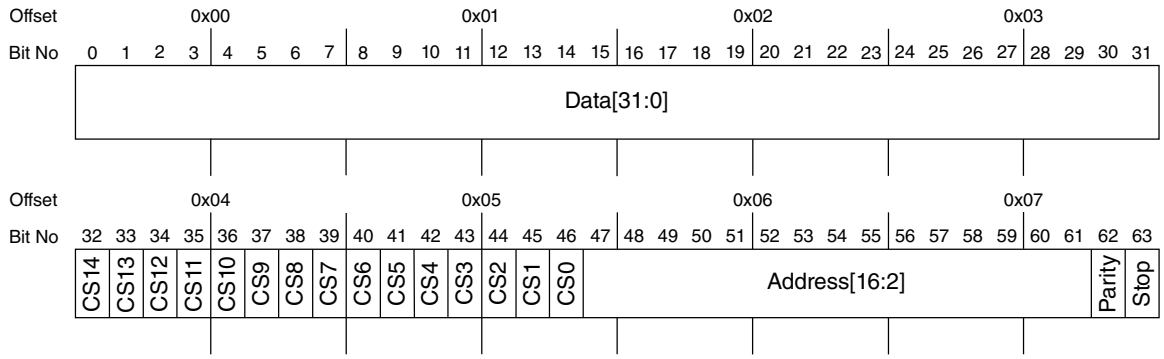


Figure 80-5. DCF record structure

Note

1. The values in the PASS_LOCK*x*PG*n* registers override the settings in the flash module's LOCK0–3 registers.
2. Values written to the PASS_LOCK*x*PG*n* registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated flash block to be locked against programming or erase.
3. The PASS_LOCK*x*PG*n* bits take effect when the life cycle is matured to "OEM Production" or older.

Example: 80.4.1.1 Implementing secure write protection

Implementing secure write protection

Implementation of secure write protection is done during initial flash setup as follows:

1. Four 256-bit passwords (passwords 0–3) must be stored in a protected OTP area of device flash called UTEST flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PG*n* –PASS_LOCK3_PG*n*) that have the same mapping of register bits to flash blocks as the flash module's SEL0–3 registers and LOCK0–3 registers.
2. DCF records must be created to define which flash memory blocks are to be locked and which blocks are not locked.

The records defined in step 2, determine the reset values of the PASS_LOCK0_PG*n*–PASS_LOCK3_PG*n* registers and therefore the write protection settings that take effect each time the microcontroller is reset.

Assume four passwords have been previously implemented and the following blocks are to be locked in the password 0 group:

Start Address	End Address	Size (KB)	Partition	LOCK*n* Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit	Read Lock group	Test Mode Disable Block Select Bit	Description
0x00FD0000	0x00FD7FFF	32	1	LOCK0	LOWLOCK[4]	20	1	3	32KB Code Flash block
0x00FD8000	0x00FDFFFF	32	1	LOCK0	LOWLOCK[5]	21	1	2	32KB Code Flash block

From the PASS section of the attached DCF_clients.xlsx:

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0000	LOCK0_PG0

Referring to Figure 80-5, the values for the DCF record are:

- Data[31:0] is 0x0030_0000 (bits 0–31) (same value used in previous examples)
- CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
- Address[16:2] is 0b000_0000_0100_0000 (bits 47–61)
- Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
- Preferred Stop value is 0 (see footnotes in Figure 80-6.

The following DCF record provides secure write protection for the blocks in Table 80-1 by setting the reset values PASS_LOCK0_PG0 to the appropriate value to lock the blocks.

Bit No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit No	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	x^1	x^2

Figure 80-6. Sample DCF record for secure write protection

The remaining blocks mapped to the PASS_LOCK0_PG*n* registers do *not* have secure write protection provided by the password 0 register set because the DATA field of DCF record has only the bits mapped to flash blocks 4 and 5 set to '1'.

CAUTION

The above example is intended to highlight the need to carefully understand how DCF records work. The protected blocks are secured by password 0 but by default they are also secured by passwords 1, 2, and 3. In contrast the blocks *not* secured by password 0 in this example *are* secured by passwords 1, 2, and 3.

See the PASS section of DCF client list table in the attached DCF_clients.xlsx.

80.4.2 Overriding secure write protection

The "PG*n*" (Password Group *n*) suffix on each the register names indicates the password each set of registers is associated with. For example, password 0, is used to unlock PASS_LOCK0_PG0, PASS_LOCK1_PG0, PASS_LOCK2_PG0, and PASS_LOCK3_PG0. The bits in the first three registers (PASS_LOCK0_PG0, PASS_LOCK1_PG0, and PASS_LOCK2_PG0) are mapped exactly the same as the bits in the flash module's SEL0–2 registers and LOCK0–2 registers.

Like the SEL0 and LOCK0 registers, the PASS_LOCK0_PG*n* registers all contain bits mapped to the low and mid address spaces, and the mapping scheme is the same. A mapped bit set to '1' in any of the registers causes the corresponding block to be locked against program or erase. A mapped bit set to '0' in any of the registers makes the corresponding block available for programming or erase.

If a bit is set to '1', locking a flash block, that value can only be changed by successfully supplying the password value for that set of registers to gain write access, and setting the bit to '0'. If the same bit is set to '1' in more than one of the PASS_LOCK*x*PG*n* register sets, each must be unlocked using the corresponding password and the bit must be set to '0' before the flash block is available for programming or erase.

Figure 80-7 shows the PASS_LOCK0_PG*n* register. As you can see, it has the same structure as the flash module's LOCK0 register. In addition to the main difference of being among a group of four parallel sets of locking registers, the other difference between the PASS_LOCK*x*_PG*n* registers and the flash LOCK*n* registers is how the reset values are determined.

By default, all mapped bits are set to '1', locking their corresponding flash blocks from programming and erase. The reset value is user-configurable, though, by writing a DCF record. This is discussed in a later section.

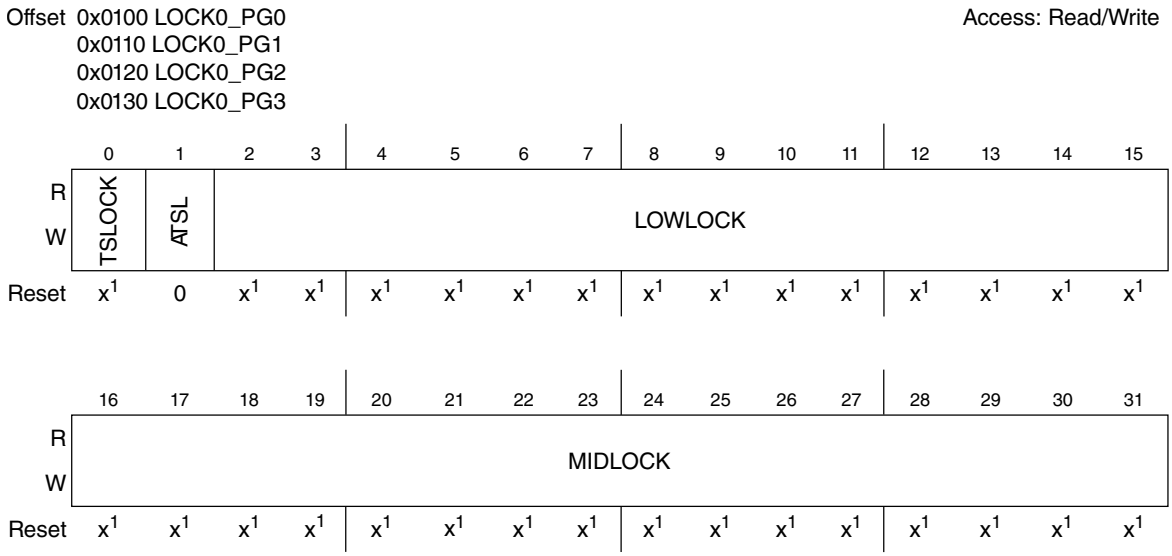


Figure 80-7. LOCK0_PG*n* register

Note

1. The values in the PASS_LOCK*x*PG*n* registers override the settings in the flash module's LOCK0–3 registers.
2. Values written to the PASS_LOCK*x*PG*n* registers do not persist through device reset. The reset value is determined by DCF record or, if no relevant DCF record has been created, each mapped bit has a reset value of '1', which causes the associated flash block to be locked against programming or erase. The purpose of the registers is to indicate secure write protection status and to provide a way to temporarily override secure write protection.
3. The PASS_LOCK*x*PG*n* bits take effect after the life cycle is matured to "OEM Production" or older.

Example: 80.4.2.1 Overriding secure write protection

Overriding secure write protection

Assume four passwords have been created but no DCF records have been written to set the reset values of the PASS_LOCK*x*_PG*n* registers, so the reset value for each mapped bit defaults to '1', causing all mapped blocks to have four levels of 256-bit password-secured write protection.

Since the settings in the PASS_LOCK*x*_PG*n* registers override the settings in the flash module's LOCK0–3 registers, the settings in those registers are irrelevant. Each of the four sets of the PASS module's LOCK registers must be unlocked by supplying the correct password for each set and then the appropriate bits must be set to '0' to make the desired flash blocks available for program or erase.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CIN*n*) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant bits of the password written to PASS CIN0.
3. Write 0xFFCF_FFFF to the PASS_LOCK0_PG0 register and verify a successful write.
4. Repeat the above steps for passwords 1–3.

Just as in previous examples, the result is the following blocks are available for program and erase operations but at a much higher level of security.

Start Address	End Address	Size (KB)	Partition	LOCK*n* Register	LOCK*n* Register Field Bit	LOCK*n* DCF Record DATA Bit	Read Lock group	Test Mode Disable Block Select Bit	Description
0x00FD0000	0x00FD7FFF	32	1	LOCK0	LOWLOCK[4]	20	1	3	32KB Code Flash block
0x00FD8000	0x00FDFFFF	32	1	LOCK0	LOWLOCK[5]	21	1	2	32KB Code Flash block

80.5 Secure read protection

Read Protection is ability of the flash controller to block read accesses to the flash memory from all bus masters. This includes the blocking of:

- Instruction fetches and load instructions from the CPU
- Debug reads from an outside tool
- DMA reads
- All other types of bus master reads

When secure read protection is active, the affected flash blocks are readable by any bus master until a debugger is attached to the JTAG port and debug is enabled. If a debugger is attached and debug is enabled while an application is running, the next access to a flash read protected region terminates with a bus error.

Most instances of flash memory read protection in the MPC5746C microcontroller are *not* configurable by the customer. An example of this is the 256-bit passwords stored in UTEST flash memory. Password-based read protection *is* customer-configurable[4] and is similar to the password-based write protection mechanism detailed in Secure write protection.

There are two significant differences between password-based read protection and password-based write protection in the MPC5746C microcontroller.

1. Read protection can only be applied to pre-defined groups of flash memory blocks. These groups, called "Read Lock groups", cannot be re-defined, meaning the blocks assigned to those groups are permanently assigned and cannot be assigned to other groups and no new groups can be created.
2. Read protection is applicable if the device is censored (Please refer PASS_LCSTAT[CNS] and PASS_LCSTAT[JUN] bit fields for details). This protection is only active either when a debugger is attached to the JTAG port and debug is enabled; or the lifecycle has been advanced to "Failure Analysis".

Another difference is that only the PASS_LOCK3_PG*n* register for each password group is used:

- PASS_LOCK3_PG0
- PASS_LOCK3_PG1
- PASS_LOCK3_PG2
- PASS_LOCK3_PG3

4. Secure read protection is user-configurable in that the default lock/unlock state for each mapped block at reset is controlled by user-created DCF records, giving the user the ability to determine the number of levels of password protection implemented. The read protection is applied to groups of blocks instead of individual blocks and the groups are not user-definable.

The next sections discuss implementing secure read protection and overriding secure write protection.

80.5.1 Implementing secure read protection

Implementation of secure read protection is done during initial flash setup as follows:

1. Four 256-bit passwords (passwords 0–3) are stored in a protected OTP area of device flash named UTEST flash. Each password secures a set of registers in the PASS module (PASS_LOCK0_PG*n* –PASS_LOCK3_PG*n*) that have the same mapping of register bits to flash blocks as the flash module's SEL0–3 registers and LOCK0–3 registers. These are the same passwords used by secure write protection and other secure functions. The passwords are only created once.
2. DCF records must be created to define which flash memory blocks are to be locked. Those records determine the reset values of the PASS_LOCK3_PG*n* registers.

   Like the other PASS_LOCK*x*_PG*n* registers, the PASS_LOCK3_PG*n* registers have two functions.

   - Reading the registers provides a lock status for each Read Locking group.
   - Changing the value of a mapped bit in a register changes the read protect status of the corresponding Read Locking group.

The layout of the PASS_LOCK3_PG*n* registers is shown in Figure 80-8. The fields relevant to implementing secure read protection are RL4–RL0.

Offset 0x010C LOCK3_PG0
0x011C LOCK3_PG1
0x012C LOCK3_PG2
0x013C LOCK3_PG3

Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PGL	DBL	MO	0	MSTR				0	0	0	RL4	RL3	RL2	RL1	RL0
W																
Reset	x[1]	x[2]	x[2]	0	1	1	1	1	0	0	0	x[2]	x[2]	x[2]	x[2]	x[2]

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	256LCK_U															
W																
Reset	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]	x[2]

Figure 80-8. PASS_LOCK3_PG*n* Register

Example: 80.5.1.1 Implementing secure read protection

Implementing secure read protection

Assume four passwords have been previously implemented.

Following is a example of a mapping of flash blocks to Read Locking groups.

Table 80-2. Sample low and mid address space flash block bit mapping

Start Address	End Address	Size	Partition	LOCKn Register[1]	LOCKn Register Field bit	LOCKn DCF Record DATA bit[2]	Test Mode Disable Block Select Bit	Description
0x00400000	0x00403FFF	16	-	LOCK 0	TSLOCK	31	-	UTest NVM Block Space[3]
0x00620000	0x0062FFFF	64	1	LOCK 0	LOWLOCK[9]	25	14	64KB HSM code block 3
0x00FF0000	0x00FFFFFF	64	1	LOCK 0	LOWLOCK[8]	24	15	64KB code block
0x00610000	0x0061FFFF	64	0	LOCK 0	LOWLOCK[7]	23	0	64KB HSM code block 2
0x00FE0000	0x00FEFFFF	64	0	LOCK 0	LOWLOCK[6]	22	1	64KB code block
0x00FD8000	0x00FDFFFF	32	1	LOCK 0	LOWLOCK[5]	21	2	32KB Code Flash block
0x00FD0000	0x00FD7FFF	32	1	LOCK 0	LOWLOCK[4]	20	3	32KB Code Flash block
0x00FC8000	0x00FCFFFF	32	0	LOCK 0	LOWLOCK[3]	19	4	32KB Code Flash block
0x00FC0000	0x00FC7FFF	32	0	LOCK 0	LOWLOCK[2]	18	5	32KB Code Flash block
0x00404000	0x00407FFF	16	1	LOCK 0	LOWLOCK[1]	17	6	16KB BAF block[4]
0x00F8C000	0x00F8FFFF	16	0	LOCK 0	LOWLOCK[0]	16	7	16 KB Code Flash Block
0x00FAC000	0x00FAFFFF	16	3	LOCK 0	MIDLOCK[7]	7	16	16KB Code Flash block
0x00FA8000	0x00FABFFF	16	3	LOCK 0	MIDLOCK[6]	6	17	16KB Code Flash block
0x00FA4000	0x00FA7FFF	16	3	LOCK 0	MIDLOCK[5]	5	18	16KB Code Flash block
0x00FA0000	0x00FA3FFF	16	3	LOCK 0	MIDLOCK[4]	4	19	16KB Code Flash block
0x00F9C000	0x00F9FFFF	16	2	LOCK 0	MIDLOCK[3]	3	20	16KB Code Flash block

Table continues on the next page...

Table 80-2. Sample low and mid address space flash block bit mapping (continued)

0x00F98000	0x00F9BFFF	16	2	LOCK 0	MIDLOCK[2[	2	21	16KB Code Flash block
0x00F94000	0x00F97FFF	16	2	LOCK 0	MIDLOCK[1]	1	22	16KB Code Flash block
0x00F90000	0x00F93FFF	16	2	LOCK 0	MIDLOCK[0]	0	23	16KB Code Flash block
0x00F84000	0x00F87FFF	16	5	LOCK1	HIGHLOCK[1]	1	38	16KB HSM Data Block 1
0x00F80000	0x00F83FFF	16	4	LOCK1	HIGHLOCK[0]	0	39	16KB HSM Data Block 0
0x01240000	0x0127FFFF	256	7	LOCK2	256KLOCK[9]	9	62	256KB Code Flash block9
0x01200000	0x0123FFFF	256	7	LOCK2	256KLOCK[8]	8	63	256KB Code Flash block8
0x011C0000	0x011FFFFF	256	6	LOCK2	256KLOCK[7]	7	48	256KB Code Flash block7
0x01180000	0x011BFFFF	256	6	LOCK2	256KLOCK[6]	6	49	256KB Code Flash block6
0x01140000	0x0117FFFF	256	6	LOCK2	256KLOCK[5]	5	50	256KB Code Flash block5
0x01100000	0x0113FFFF	256	6	LOCK2	256KLOCK[4]	4	51	256KB Code Flash block4
0x010C0000	0x010FFFFF	256	6	LOCK2	256KLOCK[3]	3	52	256KB Code Flash block3
0x01080000	0x010BFFFF	256	6	LOCK2	256KLOCK[2]	2	53	256KB Code Flash block2
0x01040000	0x0107FFFF	256	6	LOCK2	256KLOCK[1]	1	54	256KB Code Flash block1
0x01000000	0x0103FFFF	256	6	LOCK2	256KLOCK[0]	0	55	256KB Code Flash block0

1. There are multiple sets of LOCKn registers, i.e., the flash module's LOCK0–-3 registers and PASS module's PASS_LOCK0–3_PG0–3, but all use the same mapping of register bits to flash blocks. The same mapping applies the the flash module's SEL0–3 registers.
2. The DCF record DATA bit refers to the DATA[31:0] field contained in DCF records created during OEM configuration activities. The DCF records related to flash blocks are used to implement password-secured write protection, passwordsecured read protection, tamper detection, and debug port enable/disable as a permanent part of the device configuration.
3. One time programmable.
4. The Boot Assist Flash (BAF) is one time programmable and the majority of it is programmed in the factory. The BAF handles the initial start up of the device.

To provide two level of password protection for only Read Lock group 3 and leave all other groups unlocked, the RL3 bit can be set in any two of the four PASS_LOCK3_PG*n* registers. For this example, we will choose password groups 0 and 1.

We need to create DCF records that define the reset values for all RL*n* bits as 0, with the exception of PASS_LOCK3_PG0 and PASS_LOCK3_PG1. The information required is from the PASS section of the attached DCF_clients.xlsx.

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0000_1000	000_0000_0100_0011	LOCK3_PG0
000_0000_0000_1000	000_0000_0100_0111	LOCK3_PG1
000_0000_0000_1000	000_0000_0100_1011	LOCK3_PG2
000_0000_0000_1000	000_0000_0100_1111	LOCK3_PG3

Referring to Figure 80-5, the values for the DCF record are:

- For LOCK3_PG0
  - Data[31:0] is 0x0008_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_0011 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG1
  - Data[31:0] is 0x0008_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_0111 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG2
  - Data[31:0] is 0x0000_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_1011 (bits 47–61)
  - Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
  - Preferred Stop value is 0.
- For LOCK3_PG3
  - Data[31:0] is 0x0000_0000 (bits 0–31)
  - CS[14:0] is 0b000_0000_0000_1000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0100_1111 (bits 47–61)

- Parity value is irrelevant because it is not used in PASS DCF records (bit 62)
- Preferred Stop value is 0.

80.5.2 Overriding secure read protection

As noted previously, the PASS_LOCK3_PG*n* registers (see Figure 80-8) contain five read protection fields, RL0–RL4, that correspond to pre-defined Read Lock groups. To temporarily unlock a Read Lock group, the appropriate PASS_LOCK3_PG*n*[RL*n*] bits must be set to 0.

Like the register bits controlling secure write protection, the reset value of bits controlling secure read protection is determined by DCF record or, in the absence of a relevant DCF record, each bit resets to '1', causing the blocks associated with each read group to be read locked.

Example: 80.5.2.1 Overriding secure read protection

Overriding secure read protection

Building on Implementing secure read protection, where Read Lock group 3 was protected by passwords 0 and 1, temporarily overriding secure read protection for that group is accomplished by first unlocking the PASS_LOCK3_PG0 and PASS_LOCK3_PG1 registers and then setting the RL3 bit to '0' in both.

1. First write 0b00 to the PASS_CHSEL[GRP] register field to indicate registers controlled by password 0 are to be unlocked.
2. Write the 256-bit value for password 0 to the set of eight 32-bit password challenge input registers (PASS_CIN*n*) beginning with PASS_CIN0. The password must be written as a sequence of eight 32-bit writes, with the most significant bits of the password written to PASS CIN0.
3. Write a 0 to PASS_LOCK3_PG0[RL3] bit and verify a successful write.
4. Repeat the above steps for password 1.

The result of a successful sequence is that the blocks associated with the unlocked read lock group can be read by a debugger.

80.6 Debug port enable/disable

The mechanism for temporarily enabling the MPC5746C debug port after it has been locked by its life cycle state is similar to the flash memory read protection mechanism detailed in Secure read protection. The difference is that the PASS_LOCK3_PG*n*[DBL] field is used instead of the RL0–RL4 fields.

80.7 Tamper detection

The Tamper Detection Module (TDM) provides a type of flash memory write protection mechanism that forces software to write a record associated with one or more blocks in a Tamper Detection Region (TDR) before the block(s) can be erased. An additional feature is provided to enable customers to override the protection mechanism for any TDR so that no record is required to be written before a block within the TDR can be erased.

This section provides details on the mapping between DCF record bits and individual flash memory blocks to be selected for tamper detection.

See Tamper Detection Module (TDM) for details on tamper detection, including the DCF clients and two additional memory-mapped registers.

There are two significant differences between selecting blocks for tamper detection and selecting blocks for other activities, e.g., a flash erase operation:

1. Selection of blocks for tamper detection protection is done only during device configuration to customer specifications, either at the factory or by the customer during OEM production activities. The selection becomes a permanent part of the device configuration.
2. The selection of blocks for tamper detection is done by writing *DCF records*, which is why the selection becomes a permanent part of the device.

The Tamper Detect section of DCF client list table in the attached DCF_Clients.xlsx. gives the CS and Address value combinations that correspond to groups of LOCK0–LOCK3 registers, similar to the flash module's LOCK0–LOCK3 registers.

The mapping of individual flash blocks to TDM DCF record data field bits is shown in Tamper Detection Module (TDM) configuration

80.7.1 Implementing tamper detection

Implementing tamper detection requires:

1. Creating the tamper detect diary
2. Assigning blocks to Tamper Detection Regions (TDRs)

The tasks are detailed in the following sections

80.7.2 Creating the tamper detect diary

Tamper detection in the MPC5746C microcontroller is user-defined. The basic functionality provided is an erase counter, or diary, for 6 user-defined regions, called "tamper detect regions", or TDRs, within the flash memory. Each TDR is defined using a set of DCF records that is similar in function and layout to the flash module's SEL*n* and LOCK*n* registers, which contain bits matched to specific flash blocks.

For a program or erase operation to be performed on a block included in a TDR, a record must be written to a "diary" that tracks flash modification operations. The format and size of records is customer-defined. The only hardware constraints are that each diary has a maximum size of 2 KB and the minimum size of any programming operation is 8 bytes.

Note

The limiting factor for tamper detection records is the diary size. The total diary size is given by the number of TDRs (6 for MPC5746C) multiplied by 2 KB, the portion of the diary allotted to records associated with each TDR.

Since the minimum flash programming operation size is 8 bytes, the maximum number of tamper detect records for a given TDR is 256 8-byte entries.

The diary can be placed within any flash block in the flash array but to maintain the security of the TDR, there are two constraints:

- The block where the diary is placed must be assigned as OTP within the OTP registers (covered in the next section)
- The diary base address must be on a 4 KB boundary, therefore the least significant 12 bits of the base address must all be '0'

Warning

The tamper detect diary base address DCF record is a "write once" record. All subsequent tamper detect diary base address DCF records created are ignored.

Example: 80.7.2.1 Creating the tamper detect diary

Creating the tamper detect diary

If the total space required for a diary is 12 KB (as in the case of 6 TDRs), any flash block can be used because the smallest flash blocks in the MPC5746C microcontroller are 16 KB. Since the diary base address must be on a 4 KB, boundary, though, if the diary base address is within a 16 KB flash block, it can only be assigned to the base address of the block or the address given by the base address plus 4 KB.

If the base address of the 16 KB block is 0x00F90000, the base address of the diary is given by:

0b0000_0000_1111_1001_0000_0000_0000_0000 (16 KB base address)
or
0b0000_0000_1111_1001_0001_0000_0000_0000 (16 KB base address + 4 KB)

The base address of each TDR's section of the diary is:

- TDR0 = Diary_Base_Address + 0b00_0000_0000_0000
- TDR1 = Diary_Base_Address + 0b00_1000_0000_0000
- TDR2 = Diary_Base_Address + 0b01_0000_0000_0000
- TDR3 = Diary_Base_Address + 0b01_1000_0000_0000
- TDR4 = Diary_Base_Address + 0b10_0000_0000_0000
- TDR5 = Diary_Base_Address + 0b10_1000_0000_0000
- End of the diary = Diary_Base_Address + 0b11_0000_0000_0000

We need to create one DCF record that defines the base address for the diary. The information required is from the Tamper Detect section of the attached DCF_Clients.xlsx sheet.

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0001_0000	000_0000_0000_0000	Diary Base Address

Referring to Figure 80-5, the values for the DCF record are:

- Diary Base Address
  - Data[31:0] is 0x00F90000 (the base address)
  - CS[14:0] is 0b000_0000_0001_0000 (bits 32–46)
  - Address[16:2] is 0b000_0000_0000_0000 (bits 47–61)
  - Parity value is irrelevant because it is not used in Tamper Detect DCF records (bit 62)
  - Preferred Stop value is 0.

See the Tamper Detect section of DCF client list table refer to the attached DCF_Clients.xlxs sheet for details.

Note

User software must check whether the diary is full. If the diary area for a TDR is full, user software must program a DCF record to override blocking of erase from TDR, disabling tamper detect, so that erases can be done to that TDR without any diary entry. This is detailed in Overriding tamper detection.

80.7.3 Assigning blocks to Tamper Detection Regions (TDRs)

Tamper detection regions are defined by creating TDR*x*_LOCK*n* DCF records. For example, in a microcontroller with six tamper detection regions, the following records define the TDRs:

- TDR0_LOCK0, TDR0_LOCK1, TDR0_LOCK2, TDR0_LOCK3
- TDR1_LOCK0, TDR1_LOCK1, TDR1_LOCK2, TDR1_LOCK3
- TDR2_LOCK0, TDR2_LOCK1, TDR2_LOCK2, TDR2_LOCK3
- TDR3_LOCK0, TDR3_LOCK1, TDR3_LOCK2, TDR3_LOCK3
- TDR4_LOCK0, TDR4_LOCK1, TDR4_LOCK2, TDR4_LOCK3
- TDR5_LOCK0, TDR5_LOCK1, TDR5_LOCK2, TDR5_LOCK3

Each set of TDR*x*LOCK*n* DCF records mapped in exactly the same way as the flash module's SEL*n* and LOCK*n* registers (see the detailed flash block map in the flash section of the Device Configuration chapter for the mapping).

In the TDR*x*LOCK*n* DCF records:

- A bit set to '1' means the corresponding flash block is assigned to this TDR
- A bit set to '0' means the corresponding flash block is NOT assigned to this TDR

The default state (no TDRs defined) is defined as 'Block NOT assigned to the TDR'.

80.7.4 Overriding tamper detection

When a TDR's diary section is full, a Tamper Override (TO) record must be created before a block assigned to the TDR can be programmed or erased.

The details of the data section of the DCF record used to override tamper detection is shown in Figure 80-9.

warning

Bits in this DCF record is writable from 0 to 1only. Attempted writes of bits from 1 to 0 are ignored.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
W	0	0	0	0	0	0	0	0	0	0	TOE5	TOE4	TOE3	TOE2	TOE1	TOE0

Figure 80-9. Data portion of Tamper Region Override (TO) DCF record

Example: 80.7.4.1 Implementing secure write protection

Implementing secure write protection

Assume the diary Tamper Detection Region 3 is full

We need to create one DCF record that defines that overrides tamper detection for the tamper detection region whose diary is full. The information required is from the Tamper Detect section of the attached DCF_Clients.xlsx sheet.

DCF CS[14:0]	DCF address [16:2] (binary)	DCF client description
000_0000_0001_0000	000_0000_0000_0001	Tamper Region Override

Referring to Figure 80-5, the values for the DCF record are:

- Data[31:0] is 0x0000_0008 (TOE3)
- CS[14:0] is 0b000_0000_0001_0000 (bits 32–46)
- Address[16:2] is 0b000_0000_0000_0001 (bits 47–61)
- Parity value is irrelevant because it is not used in Tamper Detect DCF records (bit 62)
- Preferred Stop value is 0.

See the Tamper Detect section of DCF client list table of the attached DCF_Clients.xlsx sheet for details.

80.8 Implementing OTP

Any flash block within the flash array can be assigned, at any time, to be OTP. OTP means that flash erase of the entire block is disabled and only 64-bit double words that are already erased (i.e., flash content is 0xFFFF_FFFF_FFFF_FFFF) can be programmed. Over-programming is not possible.

Flash blocks are assigned as OTP by writing a DCF record. The block becomes OTP after the next reset.

warning

After a flash block is configured as OTP, it cannot be changed back.

This capability resides within the Tamper Detection Module (TDM). The records are detailed in the attached DCF_Clients.xlsx sheet. See the entries for the OTP_EN0, OTP_EN1, OTP_EN2, and OTP_EN3 DCF records in the Tamper Detect section of the table.

80.9 Implementing test mode disable

The MPC5746C flash memory module includes a mechanism to disable manufacturer entry into test mode. Extreme care must be taken when using this feature, as blocks that are selected to be protected in this method are able to have possible failures analyzed by manufacturer's failure analysts.

Test mode disable prevents all high voltage operations to the flash executed by the internal state-machine, as well as reads through the state machine, and reads through the Array Integrity state machine when using test mode interfaces.

Optionally, the customer can create a password to enable manufacturer entry into test mode.

80.9.1 Unconditional test mode disable seal

Warning

This process is not reversible. After completing these steps the MPC5746C microcontroller is permanently prevented from entering test mode for factory analysis.

The steps for unconditionally disabling test mode are the same steps required for implementing passcode-protected entry into test mode with one exception—the passcode specified must not be one of the following three *invalid* passcodes:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

To unconditionally disable test mode entry:

1. Select individual flash blocks to be protected by programming the appropriate values into the Test Mode Disable Block Select Group A and Test Mode Disable Block Select Group B areas shown in the UTEST flash memory map.
2. Program one of the three *invalid* passcodes shown above into the Test Mode Override Passcode area shown in the UTEST flash memory map.
3. Program the value 0x2D3C_4B5A into the Test Mode Disable Seal location specified in the UTEST flash memory map.

After the next reset is asserted, Test Mode is permanently disabled.

Note

1. Blocks selected for test mode disable can be selected in either or both Test Mode Disable Block Select Groups (A and B) because those fields are logically ORed to determine the blocks to be protected.

2. The lifecycle state must be programmed to 'Production at OEM' or later for the disable seal to be in effect.

80.9.2 Passcode-protected test mode disable seal

The steps for implementing passcode-protected entry into test mode are the same required for unconditionally disabling test mode with the exception that —the passcode specified must NOT be one of the following three *invalid* passcodes:

- 0x0000_0000
- 0xFFFF_FFFF
- 0x5555_5555

To implement passcode-protected test mode entry:

1. Select individual flash blocks to be protected by programming the appropriate values into the Test Mode Disable Block Select Group A and Test Mode Disable Block Select Group B areas shown in the UTEST flash memory map.
2. Program a valid passcode into the Test Mode Override Passcode area shown in the UTEST flash memory map.
3. Program the value 0x2D3C_4B5A into the Test Mode Disable Seal location specified in the UTEST flash memory map.

After the next reset is asserted, Test Mode is permanently passcode protected.

Note

1. Blocks selected for test mode disable can be selected in either or both Test Mode Disable Block Select Groups (A and B) because those fields are logically ORed to determine the blocks to be protected.
2. The lifecycle state must be programmed to 'Production at OEM' or later for the disable seal to be in effect.

80.9.3 Selecting flash memory blocks for test mode disable seal

Selecting flash memory blocks for test mode disable seal protection is similar to selecting blocks for tamper detection in that it involves writing to non-register flash locations. In the case of the test mode disable seal function, there are two areas of UTEST flash memory that have identical mapping to flash memory blocks. When the test mode disable seal takes effect, the mapped bits are logically ORed to determine the blocks to be protected.

CAUTION

1. The mapping of bits to blocks for this function is different from the mapping used for other functions. See the table in the device configuration chapter for the map.
2. Selecting a block for test mode disable requires a '0' in the mapped bit—not a '1' as with other functions.

The mapping of individual flash blocks to Test Mode Disable Block Select bits is shown in Table 80-3.

Table 80-3. Mapping of Test Mode Disable Block Select bitsto flash blocks

Start Address	End Address	Byte offset[1]	Test Mode Disable Block Select Bit	Description
0x00610000	0x0061FFFF	0	0	64KB HSM code block 2
0x00FE0000	0x00FEFFFF	0	1	64KB code block
0x00FD8000	0x00FDFFFF	0	2	32KB Code Flash block
0x00FD0000	0x00FD7FFF	0	3	32KB Code Flash block
0x00FC8000	0x00FCFFFF	0	4	32KB Code Flash block
0x00FC0000	0x00FC7FFF	0	5	32KB Code Flash block
0x00404000	0x00407FFF	0	6	16KB BAF block
0x00F8C000	0x00F8FFFF	0	7	16 KB Code Flash Block
		1	8	Reserved
		1	9	Reserved
		1	10	Reserved
		1	11	Reserved
		1	12	Reserved
		1	13	Reserved
0x00620000	0x0062FFFF	1	14	64KB HSM code block 3
0x00FF0000	0x00FFFFFF	1	15	64KB code block
0x00FAC000	0x00FAFFFF	2	16	16KB Code Flash block
0x00FA8000	0x00FABFFF	2	17	16KB Code Flash block

Table continues on the next page...

Table 80-3. Mapping of Test Mode Disable Block Select bitsto flash blocks (continued)

0x00FA4000	0x00FA7FFF	2	18	16KB Code Flash block
0x00FA0000	0x00FA3FFF	2	19	16KB Code Flash block
0x00F9C000	0x00F9FFFF	2	20	16KB Code Flash block
0x00F98000	0x00F9BFFF	2	21	16KB Code Flash block
0x00F94000	0x00F97FFF	2	22	16KB Code Flash block
0x00F90000	0x00F93FFF	2	23	16KB Code Flash block
		3	24	Reserved
		3	25	Reserved
		3	26	Reserved
		3	27	Reserved
		3	28	Reserved
		3	29	Reserved
		3	30	Reserved
		3	31	Reserved
		4	32	Reserved
		4	33	Reserved
		4	34	Reserved
		4	35	Reserved
		4	36	Reserved
		4	37	Reserved
0x00F84000	0x00F87FFF	4	38	16KB HSM Data Block 1
0x00F80000	0x00F83FFF	4	39	16KB HSM Data Block 0
		5	40	Reserved
		5	41	Reserved
		5	42	Reserved
		5	43	Reserved
		5	44	Reserved
		5	45	Reserved
		5	46	Reserved
		5	47	Reserved
0x011C0000	0x011FFFFF	6	48	256KB Code Flash block7
0x01180000	0x011BFFFF	6	49	256KB Code Flash block6
0x01140000	0x0117FFFF	6	50	256KB Code Flash block5
0x01100000	0x0113FFFF	6	51	256KB Code Flash block4
0x010C0000	0x010FFFFF	6	52	256KB Code Flash block3

Table continues on the next page...

Table 80-3. Mapping of Test Mode Disable Block Select bitsto flash blocks (continued)

0x01080000	0x010BFFFF	6	53	256KB Code Flash block2
0x01040000	0x0107FFFF	6	54	256KB Code Flash block1
0x01000000	0x0103FFFF	6	55	256KB Code Flash block0
		7	56	Reserved
		7	57	Reserved
		7	58	Reserved
		7	59	Reserved
		7	60	Reserved
		7	61	Reserved
0x01240000	0x0127FFFF	7	62	256KB Code Flash block9
0x01200000	0x0123FFFF	7	63	256KB Code Flash block8
		8	64	Reserved
		8	65	Reserved
		8	66	Reserved
		8	67	Reserved
		8	68	Reserved
		8	69	Reserved
		8	70	Reserved
		8	71	Reserved
		9	72	Reserved
		9	73	Reserved
		9	74	Reserved
		9	75	Reserved
		9	76	Reserved
		9	77	Reserved
		9	78	Reserved
		9	79	Reserved
		10	80	Reserved
		10	81	Reserved
		10	82	Reserved
		10	83	Reserved
		10	84	Reserved
		10	85	Reserved
		10	86	Reserved
		10	87	Reserved
		11	88	Reserved
		11	89	Reserved

Table continues on the next page...

Table 80-3. Mapping of Test Mode Disable Block Select bitsto flash blocks (continued)

		11	90	Reserved
		11	91	Reserved
		11	92	Reserved
		11	93	Reserved
		11	94	Reserved
		11	95	Reserved

1. The byte offset is from the start address of either the Test Mode Disable Block Select Group A (0x00400010) or the start address of the Test Mode Disable Block Select Group B (0x00400030) area of UTEST flash.

80.10 Security configuration planning

At minimum, designing the flash security framework consists of selecting the security features to be implemented and determining the scope of implementation for each selected feature.

In devices containing a Hardware Security Module (HSM), significant planning must go into the implementation of HSM functions. The HSM is an independent software platform within the microcontroller. It is intended to be used to implement advanced security functions that are defined by customer software.

Note

Details of advanced security features, including the advanced security architecture and the Hardware Security Module (HSM) are contained in the *MPC5746C Microcontroller Security Reference Manual*. Distribution of the document is restricted to qualified customers.

The basic security features available in the MPC5746C microcontroller include:

- Secure write protection
- Secure read protection
- Debug port enable/disable
- One-time-programmable (OTP) flash memory block assignment
- Factory test mode disable

80.10.1 Hardware Security Module (HSM)

Details of the HSM are included in the *MPC5746C Microcontroller Security Reference Manual*. It is mentioned here because the flash memory module includes blocks that can be configured for exclusive access by the HSM. If the HSM is to be used in an application it is advisable to protect HSM code by configuring it for exclusive access by the HSM. The blocks configurable for exclusive access by the HSM appear in Table 3-2 in NVM memory map. They are labeled as HSM code flash or HSM data flash blocks. They are not configured for HSM-exclusive access by default, though—they are available to act as secure flash areas but must be explicitly configured for that purpose. See the *MPC5746C Microcontroller Security Reference Manual* for details.

80.10.2 Creating password groups

Most basic security features in the MPC5746C microcontroller are password-based. Therefore, one of the early tasks that must be performed is the creation of four 256-bit passwords. They are programmed by the customer into the locations identified in the UTEST memory map (Table 3-3) in Memory Map.

The UTEST memory map fields to be written are:

- PASS Password Group 0
- PASS Password Group 1
- PASS Password Group 2
- PASS Password Group 3

Warning

The password fields are in OTP UTEST flash, so they can only be written once.

80.10.3 Planning secure write protection

Up to four levels of password-secured write protection can be assigned to any block. Note that by default, all blocks of user flash are write protected, so this could be more accurately viewed as planning which blocks of user flash will *not* have four levels of password-secured write protection. The write protection desired will depend on the usage of the flash blocks. Before creating DCF records to implement secure write protection, please thoroughly review the implementation steps detailed in Implementing secure write protection and in the attached DCF_Clients.xlsx sheet.

80.10.4 Planning secure read protection

Up to four levels of password-secured write protection can be assigned to flash blocks in pre-defined Read Locking groups. Note that by default, all blocks of user flash are read protected, so this could be more accurately viewed as planning which blocks of user flash will *not* have four levels of password-secured read protection. The read protection desired will depend on the usage of the flash blocks. Before creating DCF records to implement secure read protection, please thoroughly review the implementation steps detailed in Implementing secure read protection and in the attached DCF_Clients.xlsx shee.

80.10.5 Planning debug port enable/disable

Please see Debug port enable/disable and in the attached DCF_Clients.xlsx sheet.

80.10.6 Planning OTP flash memory block assignment

Any block of user flash can be configured as OTP. The need to configure blocks as OTP will depend on the data or code to be stored in the block. It is important to note that configuring a flash block as OTP is not reversible.

Before creating DCF records to implemented OTP flash blocks, please thoroughly review the implementation steps detailed in Implementing OTP and in the attached DCF_Clients.xlsx shee.

80.10.7 Planning factory test mode disable

Please see Implementing test mode disable and in the attached DCF_Clients.xlsx shee.

Appendix A
Release Notes

A.1 General changes

This section covers the changes, not covered below in their respective sections.

• Added *MPC5746C_Module_Reset_Status* spreadsheet in RM as attachment and a reference to this spreadsheet also listed in Module Status During Reset Process section
• Removed the redundant note "For the chip-specific implementation details of this module's instances, see the chip configuration information" from various places in RM.

• In Chapter Z2 Core description :
  • Added section End-to-End ECC Support
  • Added section End-to-End ECC fault injection by software
  • Added section End-to-End ECC Control Register 0 (E2ECTL0)
  • Added section End-to-End ECC control and configuration
  • In Table 62-7, added row for Critical Input
  • Added section Critical Input Interrupt (offset 0x00)

• In Chapter PRAMC :
  • Added P1_BO_DIS
  • Added PRI
  • Added P0_BO_DIS
  • Updated list item below Figure 70-1 : *System bus supports 64-bit data + 8-bit ECC AHB interfaces*
  • In Initialization/application information, added paragraph text *One possibility for initializing PRAMC memory space is to use a stored 64-bit word instruction such as Store Multiple Word (e_stmw) in Power Architecture VLE.*

• In Chapter Tamper Detect Module (TDM) :
  • Added Software Tamper Override Key Region (TDM_STO_KEY*n*)
  • In TDR override, added paragraph
  • In DCF clients, removed the list item *Permanently disable ability for software to permanently override one or more TDRs*
  • Updated Table 79-2
  • Added section Software Tamper Region Override Disable (STO_DIS) DCF client
  • In Tamper Region Override (TO) DCF client, updated Caution

• In Chapter IEEE 1149.7 Compact JTAG Test Access Port Controller (CJTAG) :
  • In 1149.7 TAP signals, deleted references to nTRST. It is not implemented on this chip.
  • In SP Format, deleted the list item pertaining to SScan. SScan is not implemented on this chip.
  • In Overview, deleted content pertaining to SScan. SScan is not implemented on this chip.
  • In EPU (Extended Protocol Unit) Operation, deleted the "Scan Selection Directive (SSD)" section. SSD is not implemented on this chip.

• In Chapter System Memory Protection Unit (SMPU) :
  • Updated Application information
  • Updated Region Descriptor n, Word 2 (SMPU_RD_*n*W2) name and register description

- Removed "Region Descriptor n, Word 2 Format 1 (SMPU_RD_nW2_F1)" register
- Updated Region Descriptor n, Word 3 (SMPU_RGD*n*_WRD3) description
- Updated SMPU_RGDn_WRD3[FMT], SMPU_RGDn_WRD3[ACCSET3], SMPU_RGDn_WRD3[ACCSET2] and SMPU_RGDn_WRD3[ACCSET1] as Reserved

- In Chapter Wakeup Unit (WKPU) :
  - Removed "Wakeup/Interrupt Pulldown Enable Register (WKPU_WIPDER)" register

- In Chapter Configurable Enhanced Modular IO Subsystem (eMIOS) :
  - Updated MODE
  - Updated Single Action Input Capture (SAIC) Mode
  - Removed eMIOS_OUDIS[OU23_OU0] and added OU31_OU0
  - Removed "QDEC" references from this chapter
  - Removed Notes from eMIOS_C*n*[EDPOL] bit description

- In Chapter Periodic Interrupt Timer (PIT-RTI) :
  - In Interrupts, replaced "a low-power mode" with "STOP mode", in paragraph text
  - Updated Introduction, Features with Stop mode details
  - Added Example configuration for the lifetime timer
  - Added PIT Lower Lifetime Timer Register (PIT_LTMR64L) and PIT Upper Lifetime Timer Register (PIT_LTMR64H)

- In Chapter Software Watchdog Timer (SWT) :
  - In SWT Control Register (SWT_CR), deleted the earlier reset value of this register and added a footnote
  - In SWT Control Register (SWT_CR), also added a Note in description that states "The reset value of this register is implementation specific. See the chip-specific SWT information."

- In Chapter Password and Device Security Module (PASS) :
  - Updated JTAG Password Challenge
  - Added new section Flash Memory Write and Erase Protection
  - Removed "External signal description" section

- In Chapter PLL Digital Interface (PLLDIG) :
  - Removed PLL Calibration Register 1 (PLLDIG_PLLCAL1) register

- In Chapter Enhanced Direct Memory Access (eDMA) :
  - Removed Wait row from Table 68-3
  - Updated description at Value "01" of BWC

- In Chapter Peripheral Bridge :
  - Updated Access support
  - Updated MPRA and MPRB

- In Chapter FlexCAN :
  - Added DMA and updates throughout the chapter for DMA functionality
  - The number of message buffers was changed from 64 to 96 and appropriate changes were made throughout the chapter
  - The FlexCAN module was changed to use the ISO CAN FD protocol and appropriate changes were made throughout the chapter
  - Added PREXCEN and added EDFLTDIS bits in CAN_CTRL2 register
  - Updated ERJW bit-width and description in CAN_CBT register
  - Updated FRJW bit-width and description in CAN_FDCBT register

- In Chapter Synchronous Audio Interface (SAI) :
  - I2S_TCSR[STOPE] bit is now Reserved
  - I2S_RCSR[STOPE] bit is now Reserved

- In Chapter Fault Collection and Control Unit (FCCU) :
  - Updated last paragraph in Introduction section
  - Updated description of XTMR bit
  - In FCCU_CTRL[OPR] bit, updated the field value description at value "10100"
  - Updated description in XTMR (FCCU_XTMR)

- In Chapter Decorated Storage Memory Controller (DSMC) :
  - Removed section "DSMC block diagram"
  - Removed section "DSMC timing diagram"

- In Chapter Crossbar Switch (AXBS) :

• Added "HLP" bit in AXBS_CRS*n* register • Updated reset value of "PCTL" bit in AXBS_CRS*n* register
• In Chapter Ethernet MAC (ENET) : • Deleted Note from ENET_ATCR[RESTART] bit • Deleted Note from ENET_ATCR[CAPTURE] bit • Removed section "Timer Synchronization for Multi-Port Implementations" and its reference from chapter • Updated ENET_MRBR1[R_BUF_SIZE], ENET_MRBR2[R_BUF_SIZE] and ENET_MRBR[R_BUF_SIZE] bit-width from 7 bits to 10 bits • Added TPWC bit • Updated bit field value description of TMODE bit at "1110" and "1111" values • Added Note in TCC bit description
• In Chapter Deserial Serial Peripheral Interface (DSPI) : • Added Multiple Transfer Operation (MTO) section • Added Parallel Chaining section • Added Serial Chaining section • Added "MTOE" and "MTOCNT" bits in DSI Configuration Register 0 (DSPI_DSICR0) • Updated Slave Mode section • Updated PCS4/$\overline{\text{MTRIG}}$—Peripheral Chip Select 4/Master Trigger section • Updated HT—Hardware Trigger section • Updated Features section • Updated Module signal descriptions section • Updated Deserial Serial Interface (DSI) Configuration section • Updated Change In Data Control section • Added Timed Serial Bus (TSB) section • Added MSC Dual Receiver Support with PCS Switch Over section • Updates throughout the chapter pertaining to TSB mode support • Added Interleaved TSB (ITSB) Mode section and its sub-sections • Updates throughout the chapter pertaining to ITSB mode support
• In Chapter Nexus Crossbar Multi-Master Client (NXMC) : • Removed "Peripheral interface" section and its sub sections • Removed ICTM feature and instances from this chapter • Updated Watchpoint trigger register (WT) • Updated Development control register 2 (DC2) • Updated Development control register 1 (DC1) • Added HSM master filtering and information pertaining to HSM
• In Chapter Fast OSC Digital Interface (FXOSC) : • Updated paragraph text in Oscillator startup delay • Updated Oscillator bypass mode section
• In Chapter Boot Assist Flash (BAF) : • Added information pertaining to NO_ECHO throughout the chapter
• In Chapter Self-Test Control Unit (STCU2) : • Content updated throughout the chapter

A.2 Introduction changes

• Updated Table 2-2 thoroughly • Added Table 2-3, Table 2-4 and Table 2-5 • In Table 2-6 : • In Communications row: Updated number of message buffers to 128 for FlexCAN • Updated Analog row for ADC • Updated Boot support row • Updated Safety row for MEMU instances

A.3 Memory Map changes

- In Table 3-5 :
  - Updated row corresponding to start address "0xFFF74000" and end address "0xFFF77FFF" as "Reserved".
  - In row with start address "0xFFF34000", added "TDM" region and updated Peripheral Bridge slot number as "OPACR50"
  - Replaced the BAR region row with Reserved in address range "0xFFFFC000 to 0xFFFFFFFF"
  - Several editorial updates in this table

- In Table 3-1, the row with "0x00F8C000 to 0x00FAFFFF" address is now split into 2 separate entries:
  - 0x00F8C000 - 0x00F8FFFF : 16 KB : Small HSM Code Block
  - 0x00F90000 - 0x00FAFFFF : 128 KB : Small and Medium NVM Blocks
- In Table 3-6 "Off Platform Bridge B":
  - Removed the first 6 entries for Reserved
  - Added a row in place of it, "0xF9000000 : 0xFBC17FFF : 44 MB : Reserved
- Updated MC_CGM mapped peripherals memory map section and table

A.4 Signal description changes

- In MSCR assignments, updated paragraph text from "*As an example, PAD18/PB[2] has SRE, ODE, SMC, IBE, HYS, WPS, WPUE, SSS and OBE are implemented with the reset value 0000_0000*" to "*As an example, PAD18/PB[2] has SRE, ODE, SMC, IBE, HYS, PUS, PUE, SSS and OBE are implemented with the reset value 0000_0000*".
- Added a Note : "Implemented SIUL2 Multiplexed Single Configuration Register..... and so on"
- In Table 4-1, added row for GPIO[197]

A.5 Security overview changes

- No substantial content changes

A.6 Platform overview changes

- New Chapter added in RM.

A.7 Power management overview changes

- In Accelerated Low Power Exit, updated "Required Software Configurations".

- In Introduction, added "MC_ME" and "LPU_CTL" modules to the list of Power Management infrastructure.
- In Features section, updated first bullet
- Updated Figure 7-2
- Added SRAM Configuration section

• Updated Table 7-4 • Updated Voltage monitoring section • Removed "*External regulator and low power modes*" section • Added Applications and Low Power Modes section
• Topics PORST in EXTERNAL REGULATION and External regulator contol are updated completely.

A.8 LPU changes

• In Module Disable (LPU_MDIS), description added a note "During transition from RUNx mode ..... clock cycles during RUNx to LPU transition."
• In LPU Mode Control Register (LPU_MCTL), description added a note "Software should always .....request is initiated."
• In Reset, added a note "If the application software is..... be flagged at RGM_FES[F_Z2_DBG]." • Updated Figure 8-1

A.9 Clocking changes

• Updated Figure 9-14.
• Updated Figure 9-21. • Added a note "If FXOSC failure happens, ... clock source again" in Loss of PLL lock/FXOSC clock. • Added a sentence "It is not recommended..... system clock" in FXOSC monitor
• In PLLDIG initialization information, point 1 subpoint c, changed "PLLFM[MODEN] = 1" to "PLLFM[SSCGBYP] = 0". • In Figure 9-13, updated 0101 to "LPU sys clk" and 0110 to "Z2 core clk". • In Figure 9-13, added a footnote "FXOSC_ANA_CLK is a output of analog ..... frequency as FXOSC". • In PLL Monitor, updated sentence from "There is an option to generate an interrupt when the flag bit is set" to "This will lead to FCCU_NCF[32] fault being set when the flag bit is set". • In FlexRay clocking added a sentence, "Control host interface (CHI) clock of FlexRay is sourced from FS80 group" and updated Figure 9-4

• In Peripheral Clock Restrictions added a bullet point "Flexray PE clock always need 80 Mhz..... flexray data rates".
• In Linear DFS Configuration
  ◦ Added footnote in "Case 2: Linear DFS" row for "FlexRay" in Table 9-2.
  ◦ Updated "Suggested configuration sequence for Linear DFS".

• In SWT clocking removed Note "*If SWT is configured in STOP/STANDBY/LPU modes, SIRC should be kept ON in target modes (RUNx/DRUN) also, to exit properly*". • In SWT clocking added Note "*SIRC should always be kept ON (in DRUN/RUNx modes) if at least one SWT is enabled to allow proper mode transitions*".
• Updated Figure 9-7 • Editorial updates in ENET clocking section • Updated Table 9-6 and also added new rows
• Updated "FXOSC_ANA_CLK" in Figure 9-16.
• Updated table header for Table 9-7.
• In ENET clocking updated timer clock from "40MHz" to "50MHz".

A.10 Embedded memories changes

• In SRAM Controller (PRAMC), added list items for "SRAM address spaces" for PRAMC_0 and PRAMC_1.

• PRAMC_0 valid control register bits :
  • Removed note: "This register includes reserved bits with R/W access. Values written to bits 22:25 have no effect on device function."
  • Removed R/W reserved field (bits 22-25)
  • Added PRI field (bits 22-23)
  • Added P1_BO_DIS field (bit 24)
  • Added P0_BO_DIS field (bit 25)
• PRAMC_1 valid control register bits :
  • Removed note: "This register includes reserved bits with R/W access. Values written to bits 22:25 have no effect on device function."
  • Removed R/W reserved field (bits 22-25)
  • Increased size of ROZ reserved field from 22 to 25 bits (bits 0-24)
  • Added P0_BO_DIS field (bit 25)
  • In register definitions figure, changed "PRAMC_n_PRCR1" to "PRAMC_1_PRCR1"

A.11 DCF Records changes

• In DCF records sequence :
  • Removed a paragraph, "The stop bit is never set by the factory,....."
  • Updated Table 11-4

• In Safety features of DCF clients updated sentence "Depending on its role in the device a client may be equipped with a safety feature or a combination of safety features".
• Removed *UTEST miscellaneous register [UTEST_MISC]* section from RM. Refer to attached DCF_Clients spreadsheet for all the details. • In Introduction, in third paragraph, updated the reference to attached DCF_Clients spreadsheet for details.
• Added new section External Regulation Exit Trim Wait (PMCDIG_ERETW)

A.12 Debug overview changes

• JTAG ID section, updated and optimized for clarity.
• Added section TCK clock relation with NPC clock.
• Deleted note "LPM handshake for STOP mode entry/exit has a restriction that .....STANDBY and LPU STANDBY modes." from NPC Handshake module

A.13 Reset and boot overview changes

• In Table 13-4, added a new row for SWT0. • Updated Self-Test Control Unit (STCU).

- Added new section Software Watchdog Timeout (SWT0).
- In Table 13-16, added a new row for SWT0.
- In Table 13-2, added a new Reserved row at Reset Source 11

A.14 Functional Safety changes

- Added new Chapter.

A.15 SIUL2 changes

- In SIUL2 MCU ID Register #1 (SIUL2_MIDR1) :
  - Updated PARTNUM description
  - Updated PKG description
  - Updated footnote on "PKG" field
  - Added footnotes to "MINOR_MASK" and "MAJOR_MASK" fields
- In SIUL2 MCU ID Register #2 (SIUL2_MIDR2) :
  - Updated FLASH_SIZE_1 description
  - Updated FLASH_SIZE_2 description
  - Updated PARTNUM description

A.16 Crossbar switch changes

A.16.1 Crossbar switch configuration changes

- In CRS register reset values added Note.

A.16.2 Crossbar switch module changes

- Initialization/application information :
  - Minor edit
  - Added note about configuration: other masters must be idle.
  - Added note about preventing reconfiguration.

A.17 AIPS changes

A.17.1 Peripheral Bridge configuration changes

- In Table 17-1 and Table 17-2, updated AIPS reset values.

A.17.2 AIPS module changes

- Corrected misspellings in Memory map/register definition.

- Peripheral Access Control Register (AIPS_PACR*n*) : Added note to register description regarding the PACR0 reset value.
- In register descriptions for Peripheral Access Control Register (AIPS_PACR*n*) and Off-Platform Peripheral Access Control Register (AIPS_OPACR*n*) : reworded for grammar and clarity, added paragraph about not accessing unimplemented PACR/OPACR registers, and added sentence in OPCAR register description about seeing the device's memory map for details.

A.18 DSMC module changes

- No substantial content changes

A.19 IAHBG changes

A.19.1 IAHB Gasket configuration changes

- No substantial content changes

A.19.2 IAHBG module changes

- No substantial content changes

A.20 SMPU changes

A.20.1 SMPU configuration changes

- In System Memory Protection Unit (SMPU) configuration, updated term "data storage buffer" to "store buffer" and added a paragraph at the end "HDBCR0 is the hardware debug cache control register..... reference manual for details."

A.20.2 SMPU module changes

- Application information
  - Correction: Previously, the "Modifying a region descriptor" bulleted list item at the beginning of the section stated, "Writing to Word 3 re-enables the region descriptor valid bit." Writing to RGD*n*_Word5 re-enables the region descriptor valid bit.

- In Application information section's bulleted list item "Modifying a region descriptor," clarified how to re-enable region descriptor by revising second sentence:
  - From: "Writing to RGD*n*_Word5 re-enables the region descriptor valid bit."
  - To: "Writing 1 to RGD*n*_WRD5[VLD] re-enables the region descriptor."
- In Application information section's bulleted list item "Removing a region descriptor," changed register name from RGD*n*_Word5 to RGD*n*_WRD5

A.21 SEMA42 module changes

- No substantial content changes

A.22 PCM changes

- In Bus Bridge Configuration Register 0 (PCM_IAHB_BE0) :
  - Updated register reset value to "0707_0700h"
  - Updated bit access of Reserved bits at Bit 5, Bit 13 and Bit 21 to R/W from ROZ
  - Bit[16:20], access changed to ROZ
  - Bit[29:31], access changed to R/W from ROZ

A.23 INTC changes

A.23.1 INTC configuration changes

• In Table 23-1 : Updated source description for "LPU_CTL" at Vector number 250 to "LPU_ICR[NEM] | LPU_ICR[MRIG]" from "o_interrupt".
• In Table 23-1, updated "Source Module" and "Source description" column from rows 295 to 329.
• Added note "Interrupts are not serviced during selftest" in INTC interrupt sources. • Added Interrupts handling during mode transitions section

A.23.2 INTC module changes

• No substantial content changes

A.24 WKPU changes

A.24.1 WKPU configuration changes

• In Interrupts added a note "Software must enable ..... mode of the device.".
• Updated Table 24-3 • In Pull control of wakeup pads , added a new paragraph "*The pull up/down voltage level for unconnected wakeup/ interrupt input is VDD_HV_A supply level.*"
• Added sentence "Recommended WKPU Configuration....." in Pull control of wakeup pads • Added NMI on standby exit section

A.24.2 WKPU module changes

• Added Wakeup/Interrupt Pull Enable Register (WKPU_WIPER).

A.25 PLLDIG changes

• In PLL Calibration Register 3 (PLLDIG_PLLCAL3)
  • Updated field width of MFDEN from 16 to 15 bits (bits 3..17).
  • Updated Reserved field width from 2 to 3 bits (bits 0..2).

• In Frequency modulation

• Removed equation 'Modulation depth' and corresponding text.

• Editorial updates.
• In Frequency modulation
  • Updated the first equation in the NOTE, "(PLLFM[STEPSIZE] × PLLFM[STEPNO])..."
  • Updated equation for PLLFM[STEPNO].

A.26 SIRC changes

• No substantial content changes

A.27 FIRC changes

• No substantial content changes

A.28 SXOSC changes

• No substantial content changes

A.29 FXOSC changes

• FXOSC Control Register (FXOSC_CTL)
  • Added note to register description, "The FXOSC_CTL is only writable in supervisor mode."
  • Updated note "After reset, writing the..." so "XOSC_CTL" is now "FXOSC_CTL".

• FXOSC Control Register (FXOSC_CTL)   • Updated FXOSC_CTL[I_OSC] field access to W1C.

A.30 CMU changes

• CMU Interrupt Status Register (CMU_ISR)
  • Added note to the CMU_ISR[FLLI] field description.
  • Added note to the Reserved field description at bit location 28.

• CMU Measurement Duration Register (CMU_MDR)   • Added note to MD field description, "Only values that do not cause..."
• Added a note in CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)
• CMU Interrupt Status Register (CMU_ISR)   • Updated Note in OLRI field description.
• CMU Low Frequency Reference Register CLKMN1 (CMU_LFREFR)

- Added the note "For correct synchronization of CMU_LFREFR write..." to the register description.
- CMU High Frequency Reference Register CLKMN1 (CMU_HFREFR)
  - Updated the note "For correct synchronization of CMU_HFREFR write..." to the register description.

- CLKMN1 supervisor
  - Updated NOTE "An example of determining the $HFREF_{Actual}$...".

- CMU Control Status Register (CMU_CSR)
  - Added note to CSR[SFM] field description, "MSR[MD] must be written before enabling...".

A.31 MC_CGM changes

- In Auxiliary Clock 3 Select Status Register (MC_CGM_AC3_SS) bit[7] is set to "Reserved" with RO access.
- Updated .

- In Progressive System Clock Switching, added a paragraph "It is recommended to .....SAFE mode or RESET event".

- Removed "Auxiliary Clock 3 Select Control Register (MC_CGM_AC3_SC)"
- Removed "MC_CGM Auxiliary Clock 3 Generation Overview" figure
- "This register is reset only on a POR reset", this sentence is added in the following registers:
  - Auxiliary Clock 2 Select Control Register (MC_CGM_AC2_SC)
  - Auxiliary Clock 5 Select Control Register (MC_CGM_AC5_SC)
  - Auxiliary Clock 6 Select Control Register (MC_CGM_AC6_SC)
  - Auxiliary Clock 8 Select Control Register (MC_CGM_AC8_SC)
  - Auxiliary Clock 2 Select Status Register (MC_CGM_AC2_SS)
  - Auxiliary Clock 3 Select Status Register (MC_CGM_AC3_SS)
  - Auxiliary Clock 5 Select Status Register (MC_CGM_AC5_SS)
  - Auxiliary Clock 6 Select Status Register (MC_CGM_AC6_SS)
  - Auxiliary Clock 8 Select Status Register (MC_CGM_AC8_SS)
  - Auxiliary Clock 9 Select Status Register (MC_CGM_AC9_SS)
  - Clockout 1 Select Register (MC_CGM_CLKOUT1_SC)
  - PCS Switch Duration Register (MC_CGM_PCS_SDUR)
  - PCS Divider Change Register 1 (MC_CGM_PCS_DIVC1)
  - PCS Divider Change Register 2 (MC_CGM_PCS_DIVC2)
  - PCS Divider End Register 1 (MC_CGM_PCS_DIVE1)
  - PCS Divider Start Register 1 (MC_CGM_PCS_DIVS1)
  - PCS Divider End Register 2 (MC_CGM_PCS_DIVE2)
  - PCS Divider Start Register 2 (MC_CGM_PCS_DIVS2)
  - Auxiliary Clock 9 Select Control Register (MC_CGM_AC9_SC)
  - System Clock Select Status Register (MC_CGM_SC_SS)
- "This register is reset only on POR reset, while .... corresponding to programmed value", this sentence is added in the following registers:
  - Aux Clock 6 Divider Configuration 0 for CLKOUT_0 (MC_CGM_AC6_DC0)
  - CLKOUT1_DC Configuration Register (MC_CGM_CLKOUT1_DC)
  - System Clock Divider 0 Configuration Register (MC_CGM_SC_DC0)
  - System Clock Divider 1 Configuration Register (MC_CGM_SC_DC1)
  - System Clock Divider 2 Configuration Register (MC_CGM_SC_DC2)
  - System Clock Divider 3 Configuration Register (MC_CGM_SC_DC3)
  - System Clock Divider 4 Configuration Register (MC_CGM_SC_DC4)
  - System Clock Divider 5 Configuration Register (MC_CGM_SC_DC5)
  - System Clock Divider 6 Configuration Register (MC_CGM_SC_DC6)
- "This register is reset only on a PHASE3 reset", this sentence is added in the following registers:
  - Divider Update Abort Register (MC_CGM_DIV_UPD_ABRTD)
  - System Clock Divider Ratio Change Register (MC_CGM_SC_DIV_RC)
  - Divider Update Type Register (MC_CGM_DIV_UPD_TYPE)
  - Divider Update Trigger Register (MC_CGM_DIV_UPD_TRIG)
  - Divider Update Status Register (MC_CGM_DIV_UPD_STAT)

A.32 Input Glitch Filter Changes

A.32.1 IGF module changes

- No substantial content changes

A.33 HFIF Changes

A.33.1 HFIF chapter changes

- No substantial content changes

A.34 ADC changes

A.34.1 ADC configuration changes

- Added new section: ADC's stop-stop ack protocol

A.34.2 ADC module changes

- Added subsection Conversion time for BCTU interface to section Conversion time.

- Updated Reserved field access from RW to ROZ (bit 16) in Interrupt Mask Register (ADC_IMR) .
- Updated the access of Reserved fields to ROZ in Conversion Timing Register 0 (ADC_CTR0) .

- Editorial changes

- Main Status register (ADC_MSR)
  - Added the sentence "MSR[CHADDR] is only valid..." to the MSR[CHADDR] field description.

- BCTU Control mode
  - Removed the paragraph "If another BCTU trigger is received..."

- In DMA functionality
  - Updated content throughout.

- Calibration, BIST Control and status Register (ADC_CALBISTREG)
  - Updated NOTE in the NR_SMPL field description.

- Updated reset values for the following registers:
  - Conversion Timing Register 0 (ADC_CTR0)

- Conversion Timing Register 1 (ADC_CTR1)
- Conversion Timing Register 2 (ADC_CTR2)

A.35 CMP changes

A.35.1 CMP configuration changes

- No substantial content changes

A.35.2 CMP module changes

- No substantial content changes

A.36 BCTU changes

A.36.1 BCTU configuration changes

- Added new section: BCTU's stop-stop ack protocol

A.36.2 BCTU module changes

- In Introduction and BCTU overview :
  - Updated number of trigger signals information.
  - Updated number of ADC modules information.
- In Introduction and BCTU Interacting with other peripherals removed information related to eTPU.
- Removed BCTU_SFTRGR3 register.

- In Trigger priority and multiplexing , updated Figure "Priority stages architecture" to show correct number of ADCs.

A.37 eMIOS changes

A.37.1 eMIOS configuration changes

- Updated Figure 37-1 : Marked the Red bus as Counter_bus_F.

A.37.2 eMIOS module changes

• No substantial content changes

A.38 MC_PCU changes

• No substantial content changes

A.39 PMCDIG changes

• In PMCDIG_MCR[LP_ST_CHG_IN_PRGRS] register, added a sentence, "This bit should always be set to '0' before a low power mode transition request is initiated".
• In SGSR register, added a note, "Bits 24-30 will remain set and will not get cleared if respective supply is below voltage monitor threshold. Please refer device data sheet for voltage monitor threshold levels.".

A.40 MC_ME changes

• Updated bitfield description of S_MRIG bit in Invalid Mode Transition Status Register (MC_ME_IMTS)
• In Use Case 1: Software Tasks with Different Clock Requirements, added a note "All ME_PCTLx controlled ..... register configuration."
• In Mode Control Register (MC_ME_MCTL), added a note "Software should always.....request is initiated".
• In Example 1: Minimal Code to Bring Up a PLL, updated code.

• In Memory Map and Register Definition, added a note "All registers are read only in user mode".
• In S_FXOSC bit of Global Status Register (MC_ME_GS) added a note "S_XOSC alone should not be relied on..... is 1 and OLRI flag = 0".
• In Mode Enable Register (MC_ME_ME) updated modes in the description. RESET changed to RESET_FUNC and RESET_DEST is added.
• In STOP0 Mode added notes:
  • If waking up from stop..... clearing module interrupt.
  • All interrupts that are ..... which causes a hang.
  • A potential system hang scenario ..... finished before stop mode change.
• In Protection of Mode Configuration Registers added three bulet points, "If the FIRC_CLK is selected....."

• Updated Figure 40-5
• Updated reset value of Global Status Register (MC_ME_GS) to 0x0C13_0010h.

A.41 GPR changes

• Updated reset value of GPR Status Register (GPR_STAT) to 0x0000_0002. • Access changed to RO (Read only) for bit 30 of GPR Status Register (GPR_STAT)
• Bit descriptions and bitfield value description are updated for CMP2_STDBY, CMP1_STDBY, CMP0_STDBY of GPR Control Register (GPR_CTL)
• In GPR Control Register (GPR_CTL), GPR_CTL[GPR_HSM_RAM_FAST_INIT_BYPASS] is placed at GPR_CTL[0].

A.42 DSPI changes

A.42.1 DSPI configuration changes

• Added new section: DSPI's stop-stop ack protocol
• Added TSB mode information in Table 42-1 and Table 42-2.

A.42.2 SPI configuration changes

• Added TSB information in Table 42-3

A.42.3 DSPI module changes

• In Status Register (DSPI_SR) - Changed TXRXS bitfield access from W1C to RO.

• Deleted note from SOUT—Serial Output.
• Clarified Memory Map/Register Definition introductory text by changing second sentence:
  • From: "Write access to the POPR and RXFRn also results in a transfer error."
  • To: "Any Write access to the POPR and RXFRn also results in a transfer error."

• Updated PUSH TX FIFO Register In Master Mode (DSPI_PUSHR) description.
• In section, PCS0/$\overline{SS}$—Peripheral Chip Select/Slave Select : Added note.
• In section, Memory Map/Register Definition : Added note
• Updated section, Receive FIFO Drain Interrupt or DMA Request with, "Configure the DMA to drain only one FIFO location per transfer". • Updated notes in Command FIFO Fill Interrupt or DMA Request and Transmit FIFO Fill Interrupt or DMA Request • Updated bit field descriptions for SR[TFFF, RFDF] • Updated bit field descriptions for SR[CMDFFF] • Updated notes in Command FIFO Fill Interrupt or DMA Request and Transmit FIFO Fill Interrupt or DMA Request • Updated bit field descriptions for SR[TFFF, RFDF] • Updated bit field descriptions for SR[CMDFFF]
• In Status Register (DSPI_SR), edited field description of:

• CMDFFF • RFDF • TFFF
• In Memory Map/Register Definition Note, edited text and reorganized items as a list
• In Status Register (DSPI_SR), edited field description of: • CMDFFF_DIRS

A.43 ENET changes

A.43.1 ENET configuration changes

• Added new section: ENET's stop-stop ack protocol • Added new section: Recieve Parser Match Array Table Depth

A.43.2 ENET module changes

• In Ethernet Control Register (ENET_ECR), 20-bit Reserved field was divided into two fields and their corresponding description's were added.
• Frame parser : Added “Initialize the entire parser memory to zero before creating entries in parser entry table.” to section. • Ethernet interfaces : Removed "ECR[SPEED]" column from table. • Adjustable Timer Control Register (ENET_ATCR) : • Added note to register description regarding when fields can be written. • Changed access of reserved bits 19 and 21 from ROZ to RW. Changed field descriptions of these bits. • Ethernet Control Register (ENET_ECR) : Reworded descriptions, but not meaning, of reserved fields.
• Frame parser : In "Receive parser flowchart" figure, changed "Freescale Semiconductor" to "NXP B.V." • Legacy buffer descriptors : Changed first paragraph from "To support the Ethernet controller on previous Freescale devices, legacy FEC buffer descriptors are available. To enable legacy support, clear ENET*n*_ECR[1588EN]." to "To support the Ethernet controller on previous chips, legacy FEC buffer descriptors are available. To enable legacy support, write 0 to ENET*n*_ECR[1588EN]."

A.44 FlexCAN changes

A.44.1 FlexCAN configuration changes

• Added new section: External time tick
• Added new section: FlexCAN's stop-stop ack protocol

A.44.2 FlexCAN module changes

• Modified description of CRCERR and STFERR. Changed "in either a non-FD message or in the slow rate part of a FD message" to "in a non-FD message or else in an FD message's arbitration or data phase." • Modified description of BIT1ERR and BIT0ERR. Changed "Arbitration Phase" to "arbitration or data phase."
• Added FlexCAN message buffer memory map.
• In the introduction section of Memory map/register definition, added text explaining that there is a separate message buffer memory map section, and added a link to that section.
• Table in CAN_CTRL2[RFFN] field description was modified to show that "Remaining Available Mailboxes" end value is 95.
• Removed reference to Doze mode in CAN_MCR[PNET_EN] field description. • Removed reference to Doze mode in Modes of operation.
• In Protocol timing, in bullet point about Time Segment 1 changed "4 to 16 time quanta" to "2 to 16 time quanta" and changed "1 to 39 time quanta" to "2 to 39 time quanta." • In Figure 44-6, changed "4...16" to "2...16" for Time Segment 1. • In Figure 44-7, changed "4...96" to "2...96" for Time Segment 1 in top image. Removed references to triple sampling in bottom image.
• In Transmission abort mechanism, added line to abort procedure summary list about reconfiguring EDL, BRS, and ESI fields in MB.
• Editorial cleanup throughout chapter.
• Modified description of Rx Individual Mask Registers (CAN_RXIMR*n*) to note option of using RXIMR memory region as general access memory. • Modified Bus interface to explain how to use RXIMR memory region as general access memory.

A.45 I2C changes

A.45.1 IIC configuration changes

• Added new section: I2C configurations

A.45.2 I2C changes

• Moved information from Clock rate and IBFD settings into two new subsections, Timing definitions and Divider and hold values, and revised this information. • In I2C Bus Frequency Divider Register (I2C_IBFD), revised the IBC field description to match the changes in Clock rate and IBFD settings. • In START condition and STOP condition, added "See Clock rate and IBFD settings and I2C Bus Frequency Divider Register (I2C_IBFD) for the associated timing requirements."
• Editorial changes.

A.46 LinFlexD changes

A.46.1 LinFlexD configuration changes

• Added new section: LinflexD's stop-stop ack protocol

A.46.2 LINFlexD module changes

• In section: Memory map and register description: Updated list item: 'If not specified ... '
• In section: Memory map and register description: Removed: "If generic slave = 0 ... cannot be programmed." from LINCR2[IOPE]
• In section: Memory map and register description: Updated the access of bits 27 and 28 from RW to ROZ in register LINSR • In section: Memory map and register description: Updated the bitfield description of bit TDFL_TFC from 'Bit 15 is reserved ... ' to 'Bit 16 is reserved ... '
• In section: Memory map and register description: Updated the access and description of bit: 25 of LINCR1 register to RW. • Updated the description of LINSR[LINS] for Init mode.
• Minor editorial updates throughout
• Changed 'Adsorb' to 'Absorb' in UART – TX mode and UART – RX mode
• In section: Memory map and register description: Updated the note 'For LIN mode, in case a bit error ...' present in bitfield DTF

A.47 FlexRay changes

A.47.1 FlexRay configuration changes

• No substantial content changes

A.47.2 FlexRay module changes

• Redefined StopWatch Count Register (FR_STPWR) as StopWatch Count High Register (STPWHR) and StopWatch Count Low Register (STPWLR).
• Updated bitfield description of FID_EF.
• Updated slot related conditions in Slot Status Counter Registers

A.48 SAI changes

A.48.1 SAI configuration changes

• Added new section: SAI's stop-stop ack protocol

A.48.2 I2S/SAI changes

• No substantial content changes

A.49 PIT-RTI changes

A.49.1 PIT-RTI configuration changes

• Updated paragraph text in Input Output section • In PIT_14 row, added FlexCAN_0 in Table 49-1 • In PIT_15 row, added FlexCAN_4 in Table 49-1
• Added new section: PIT-RTI's stop-stop ack protocol

A.49.2 PIT module changes

• No substantial content changes

A.50 SWT changes

A.50.1 SWT configuration changes

• In SWT register reset value, updated reference to reset value of SWT_0 Control register (SWT_0_CR).
• Updated SWT register reset value : added SWT_HSM_TO and SWT_HSM_CR register reset values. • Added new section: Service mode watchpoint input.

A.50.2 SWT module changes

- In description of each MAP*n* field in SWT Control Register (SWT_CR)
  - Added new paragraph beginning with this sentence: "The number of this field corresponds to the logical ID of the bus master."
  - Changed last paragraph:
    - from: "Not all MAPn fields are implemented. See the device configuration section for master ports that are implemented on the crossbar switch."
    - to: "For the masters with the logical IDs to which the MAP*n* fields correspond, see the chip-specific information for the Crossbar Switch."

- In "Memory Map and Registers"
  - In introductory text, incorporated minor edits and reorganized some paragraph text as list
  - Incorporated minor edits in each register description
- In SWT Control Register (SWT_CR) description of ITR field
  - In Note, changed "reset requests" to "reset requests and interrupt requests"
- In SWT Control Register (SWT_CR) description of RIA field
  - Added a Note: "For a description of how this chip implements SWT reset requests resulting from SWT invalid accesses, see the chip-specific SWT information."
  - Edited description of field value 1
- In Window mode edited and reorganized entire section

A.51 RTC-API changes

A.51.1 RTC-API configuration changes

- Added Seamless RTC operation across functional and destructive resets
- Added new section: RTC-API's stop-stop ack protocol

A.51.2 RTC-API module changes

- In RTC Compare Value register (RTC_RTCVAL) register description added sentence "The minimum value of RTCVAL should be 4".

- Throughout this chapter: Removed low-power related information
- In Features, removed bullet item: *Operates in all modes of operation*
- Removed following sections:
  - Test Mode
  - Detailed signal description
- Removed *as the ipg_debug is detected active* from FRZEN description
- Replaced *assertion of trigger_en_out signal* with *trigger* in TRIG_EN description
- Updated ANLCMP description
- In Functional Mode, removed phrase *and the input isolation is disabled*
- In RTC :
  - Removed *ipg_clk*
  - Replaced *wakeup_signal* with *wakeup*
  - Replaced *trigger_en_out signal* with *trigger signal to Analog Comparator*

A.52 STM changes

A.52.1 STM configuration changes

- No substantial content changes

A.52.2 STM module changes

- No substantial content changes

A.53 MC_RGM changes

- In 'Functional' Event Status Register (MC_RGM_FES), bitfieldvalue description of "1" in F_JTAG_FUNC bit is changed to "A JTAG Functional Reset event has occurred. For Example, the EXTEST, HIGHZ, and CLAMP instructions cause a JTAG 'functional' reset event to occur, which sets this field to 1"

- Editorial change: In 'Functional' Bidirectional Reset Enable Register (MC_RGM_FBRE) updated sentence *from* "This register is reset only on power-onThis register is reset only on power-on and any ‘destructive’ reset" *to* "This register is reset only on power-on and any ‘destructive’ reset".
- In Table 53-1, updated "What gets Reset" column for external and functional resets.

- Updated Figure 53-1

- In RGM_memoryMap added notes "Reserved fields ..... reset values only." and "The following registers are reset to their default values.....".
- In External Reset bullet point added "a ‘functional’ reset event configured.... reset"

- In 'Functional' Bidirectional Reset Enable Register (MC_RGM_FBRE) :
  - Updated register reset value to 0x0003_7FBC.
  - Updated bit descriptions and access for various bits.

A.54 SSCM changes

A.54.1 SSCM configuration changes

- No substantial content changes

A.54.2 SSCM module changes

• "Freescale production" is changed to "MCU Production" in following topics:
  • In bitfield value description of 110 in LC.
  • Life Cycle.
  • Table 54-13
• In Figure 54-3 updated "Freescale production" to "MCU Production".

• In Memory map and register definition added a note "Reserved IPS register access is prohibited"

A.55 BAF changes

A.55.1 BAF configuration changes

• MPC5746C cores/BAF CPU*n* :
  • Edited first sentence for clarity
  • Added space between device name and "core" in table column heading

• Added section BAF NMI event handling. • In Table 55-8, removed rows for "SC_SS" and "SWT_0_CR".

A.55.2 BAF module changes

• Flow of control :   • Added footnote to Figure 55-1 : "SWT0 is disabled after serial boot regardless of the configuration specified in the UTEST_MISC[SWT0_WDG_DIS] register field."
• Introduction :   • In second paragraph, "...services the watchdog and CSE..." changed to "...services the watchdog and HSM..."
• BAF NMI Event Handling :   • Topic has been moved to the BAF configuration section

• DATA32 – 32-bit data DCF client :
  • Phrase, "masks the least significant 2-bits to ‘0’", removed from sentence, "BAF takes the memory address provided by the ADDR client, masks the least significant 2-bits to ‘0’ (Only 32-bit aligned writes are supported) and writes this 32-bit data to the address."
• DATA16 – 16-bit data DCF client :
  • Phrase, "masks the least significant 2-bits to ‘0’", removed from sentence, "BAF takes the memory address provided by the ADDR client, masks the least significant bits to ‘0’ (Only 16-bit aligned writes are supported) and writes least significant 16-bit data to the address."

• Serial boot mode protocol :
  • Removed following statement: "If exiting from Standby mode, any core can be used to start execution"
• Flow of control :
  • Figure updated: In final decision block, "Low Power Exit Status" changed to "Low Power Standby Exit Status"

• Serial boot mode protocol :   • Note added: "Serial boot is not allowed from standby mode exit."
• Optionally perform serial boot :   • In first paragraph, changed "JDP/FSL Production" to "MCU Production".

- Boot modes :
  - In second bulleted item, changed "JDP/FSL Production" to "MCU Production".
- Flow of control :
  - In Figure 55-1, changed "Lifecycle == JDP/FSL_Production" to "Lifecycle == MCU_Production"

- Introduction :
  - Added two paragraphs describing initial task of checking for DCF record ECC errors and clearing ECC errors latched during SSCM processing.
- Flow of control :
  - In Figure 55-1, added processing block and decision block at beginning of flow for clearing errors latching during SSCM processing and checking for uncorrectable ECC error when reading DCF records starting address.

- Device initialization :
  - In item beginning with, "BAF checks for multiple cores in execution, changed second sentence from, "...then device is put in SAFE mode and made to wait for an external event", to, "...then the device is put in SAFE mode and executes a wait instruction in a loop".
  - In the item beginning with, "BAF sets up the exception vector table...", first sentence changed to, "...which may occur if there are any uncorrectable ECC errors in the BAF execution path (was, "...which may occur if there are any uncorrectable ECC errors.")
  - In the item beginning with, "BAF sets up the exception vector table...", changed, "If an ECC error occurs during reading of DMEM, the BAF will still issue a destructive reset.", to, "If the machine check exception occurs due to any other reason during BAF execution, the BAF will still issue a destructive reset."

A.56 NXMC changes

A.56.1 NXMC configuration changes

- No substantial content changes

A.56.2 NXMC module changes

- For Development control register 2 (DC2) Seperated bit field 0 from [23:0] and added 'Read/Write – default values must not be changed for normal operation' in description.
- For Development control register 2 (DC2) added 'This is read/write field, so default value must not be changed changed for normal operation' for EWC field value description.
- For Development control register 1 (DC1) added 'Read/Write – default values must not be changed for normal operation' for field 31-29 in description.

A.57 JTAG changes

A.57.1 JTAGC configuration changes

- In JTAGC Control 1 register : TMNXMC[12:13] bit is marked as Reserved.

A.57.2 JTAGC module changes

- No substantial content changes

- In Table 57-2 of External signal description :
  - Moved content from Pull column to Reset State column and removed Pull column
  - Updated "Up" to "Weak pullup"

A.58 JDC changes

- In Use case example :
  - Added new step #5 (between previous steps #4 and 5) "Check if a Flash program/erase operation is in progress. If a Flash program/erase operation is in progress, wait for it to complete".
  - Renumbered subsequent steps.

A.59 CJTAG changes

- No substantial content changes

A.60 NPC changes

A.60.1 NPC configuration changes

- Added sections:
  - EVTO sharing - Pulse time
  - Trace messages for transactions with error responses
  - Trace loss during resets
  - EVTI_IN pin

A.60.2 NPC module changes

- Minor editorial change in IEEE 1149.1-2001 (JTAG) TAP.

A.61 Core Complex Overview changes

- No substantial content changes

A.62 Z2 Core description module changes

A.62.1 Core description module changes

- No substantial content changes

A.63 Z4 Core description module changes

- No substantial content changes

A.64 Z2 Core debug support module changes

A.64.1 Core Debug Support changes

- No substantial content changes

A.65 Z4 Core Debug Support changes

- No substantial content changes

A.66 Z2 Nexus 3 module changes

A.66.1 Core Nexus Module changes

- In sections Program Trace Synchronization messages and Data Trace Synchronization messages, in the fourth bullet for both sections, added phrase "or suppression".

A.67 Z4 Nexus 3 module changes

A.67.1 Z4 Core Nexus Module changes

- No substantial content changes

A.68 eDMA module changes

- Error Status Register (DMA_ES) : In bullet about uncorrectable TCD SRAM errors, specified name of bit.
- eDMA initialization : In bullet beginning with "Enable any hardware service requests..." replaced "ERQH and ERQL registers" with "ERQ register."
- Error Status Register (DMA_ES) : In description, replaced "See the Error Reporting and Handling section" with "See Fault reporting and handling.

A.69 DMACHMUX changes

A.69.1 DMACHMUX configuration changes

- No substantial content changes

A.69.2 DMAMUX module changes

- No substantial content changes

A.70 PRAMC changes

A.70.1 PRAMC configuration changes

- No substantial content changes

A.70.2 PRAMC module changes

- No substantial content changes

A.71 Flash memory controller changes

A.71.1 Flash memory controller configuration changes

- Added new section : Master IDs

A.71.2 Flash memory controller module changes

- Flash memory controller memory map :
  - Revised content describing possible non-deterministic behavior caused by updating flash memory controller programming model concurrently with a flash write operation. Added description of workaround enabling software to invalidate port read buffers without updating programming model.

- For clarity, removed "This bit is cleared by hardware reset" or "This field is cleared by hardware reset" from descriptions of P*n*_M*m*PFE, P*n*_DPFEN, P*n*_IPFEN, and P*n*_PFLIM in:
  - Platform Flash Configuration Register 1 (PFLASH_PFCR1)
  - Platform Flash Configuration Register 2 (PFLASH_PFCR2)
  - Platform Flash Configuration Register 4 (PFLASH_PFCR4)

  These bitfields can also be cleared by writing 0.

- Flash memory controller memory map :
  - Typo fixed: "If application software functionality includes invalidating a port read buffer, t recommended method..." changed to, "If application software functionality includes invalidating a port read buffer, the recommended method..."

- ECC on data flash accesses :
  - Correction: ECC errors are reported to the MEMU (previously stated errors were suppressed from being reported).
  - Content significantly rearranged.

A.72 Embedded Flash Memory changes

A.72.1 Embedded Flash Memory configuration changes

- Start and end address is updated in the following register bit mapping:
  - C55FMC_LOCK0 and C55FMC_SEL0 register bit mapping
  - C55FMC_LOCK1 and C55FMC_SEL1 register bit mapping
  - C55FMC_LOCK2 and C55FMC_SEL2 register bit mapping
- Added note "Do not access the reserved area of the C55FMC memory map" in Embedded Flash Memory register reset value

A.72.2 c55fmc module changes

- No substantial content changes

A.73 MEMU changes

A.73.1 MEMU configuration changes

- In Table 73-5, Table 73-6, Table 73-7, Table 73-8, and Table 73-9, added a column "Clock relationship with respect to MEMU".
- In Concurrent overflow register (OFLW) implementation for MEMU_0, updated note: *from* "As per this table the ..... the FLASH Error." *to* "This table shows that..... error and a RAM uncorrectable error.".

A.73.2 MEMU changes

- In System RAM uncorrectable error reporting table address register (MEMU_SYS_RAM_UNCERR_ADDR), in the ERR_ADD field description, deleted "This field must be read-only when its corresponding valid bit (SYS_RAM_UNCERR_STS[VLD]) = 0."
- In Peripheral RAM uncorrectable error reporting table address register (MEMU_PERIPH_RAM_UNCERR_ADDR), in the ERR_ADD field description, deleted "This field must be read-only when its corresponding valid bit (PERIPH_RAM_UNCERR_STS[VLD]) = 0."
- In Flash memory uncorrectable error reporting table address register (MEMU_FLASH_UNCERR_ADDR), in the ERR_ADD field description, deleted "This field must be read-only when its corresponding valid bit (FLASH_UNCERR_STS[VLD]) = 0."
- In Design overview, added an example of how MEMU processes a 0-2-2 sequence.

A.74 FCCU changes

A.74.1 FCCU configuration changes

• Added topic FCCU_CFG register event bit values by source (N and C) • Updated reset value of FCCU_CFG register in Table 74-1

• In Table 74-2 :
  • Updated details in rows for NCF[0], NCF[1], NCF[2] and NCF[3].
  • Updated description column for NCF[5], NCF[8], NCF[10], NCF[11], NCF[69] and NCF[70].
  • Added footnote "*These faults may get set during chip level debug mode and needs to be ignored*", to NCF[2], NCF[3], NCF[25], NCF[26], NCF[34], NCF[36], NCF[37], NCF[38], NCF[39], NCF[40], NCF[42], NCF[43], NCF[44], NCF[51], NCF[53] and NCF[55].

A.74.2 FCCU module changes

• In the Mode Controller Status (FCCU_MCS) register description, changed the footnote for the register reset value from *The MC_ME module changes the value of the register immediately after the chip leaves RESET mode.* to *The reset value is chip-specific. See the chip-specific FCCU information. (The MC_ME module changes the value of the register immediately after the chip leaves RESET mode.)*.
• In the NA Freeze Status (FCCU_N2AF_STATUS) register description, in the NAFS field description, deleted this note: *This field can be read and written but writing 1 has no effect.* • In the AF Freeze Status (FCCU_A2FF_STATUS) register description, changed the access types of the AF_SRC and AFFS fields from read-write to read-only. In the AF_SRC and AFFS field descriptions, deleted this note: *This field can be read and written but writing 1 has no effect.* • In the NF Freeze Status (FCCU_N2FF_STATUS) register description, changed the access types of the NF_SRC and NFFS fields from read-write to read-only. In the NF_SRC and NFFS field descriptions, deleted this note: *This field can be read and written but writing 1 has no effect.* • In the FA Freeze Status (FCCU_F2A_STATUS) register description, in the FAFS field description, deleted this note: *This field can be read and written but writing 1 has no effect.*
• In the Noncritical Fault State Configuration (FCCU_NCFS_CFG*n*) and NMI Enable (FCCU_NMI_EN*n*) register descriptions, removed the following inaccurate note: *Do not configure the same channel for both a RESET reaction in the FCCU_NCFS_CFGn register and an NMI reaction in the FCCU_NMI_ENn register at the same time.*
• In Use cases and limitations, rewrote Configuration guidelines (formerly titled "Misconfigurations") to improve the accuracy and clarity of the material by putting it in terms of guidelines you must follow. Added guidelines about what you must do if you enable chip reset as the type of Fault-state reaction for a noncritical fault channel and if you enable Alarm state for the channel.
• In Fault recovery, added the word *noncritical* before the word *FAULT*.
• In the PhysicErrorPin field description, added the spelled-out name of the field, Fault Output (EOUT) States. Replaced *PhysicErrorPin : Physical Error Pin's state. PhysicErrorPin[0] corresponds to error pin 0 and PhysicErrorPin[1] corresponds to error pin 1. For each of these, the bit values have the following meanings: 0: Error Pin is at logical 0; 1: Error Pin is at logical 1* with *Applies only after successfully running an OP3 operation (OPR) and only when the EOUT signals are active (FCCU_SET_AFTER_RESET). Indicates the states that FCCU is driving on the EOUT signals.* Deleted *During Fault state, a static or toggling value is observed based on selected protocol.* Replaced the list of two values and meanings with a list of four values and meanings.
• In the Mode Controller Status (FCCU_MCS) register description, changed the general description from *and whether FCCU was in Fault state when FCCU captured each mode* to *and whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured each mode*. In the FS0 field description (and similarly in the FS1, FS2, and FS3 field descriptions), changed *Indicates whether FCCU was in Fault state when*

FCCU captured MCS0 to *Indicates whether FCCU was in Fault state and NMIOUT was asserted (as the result of a noncritical fault) when FCCU captured MCS0*; changed the meaning for the value 0 from *A state other than Fault state* to *Not in Fault state or NMIOUT not asserted*; changed the meaning for the value 1 from *Fault state* to *In Fault state and NMIOUT asserted*.

- In the Permanent Configuration Lock (FCCU_PERMNT_LOCK) register description, changed the register name from *Permanent Lock* to *Permanent Configuration Lock*; in the general description, changed *The FCCU_PERMNT_LOCK register is used for locking configuration permanently by writing FFh. This register is available only in supervisor mode. The permanent lock can only be removed by applying a reset (not necessarily power on reset). This register is write only.* to *Writable only by code running in Supervisor mode. Permanently locks the configuration, which prevents FCCU from entering Configuration state until FCCU is reset.*; changed the access type of the reserved field from Read-Only Zero to Reserved, Unimplemented. In the PERMNTKEY field description, changed *Transition Locking Key value* to *Permanent Configuration Lock—Writable only by code running in Supervisor mode. Permanently locks the configuration, which prevents FCCU from entering Configuration state until FCCU is reset.*; changed the meaning for the value FFh from *Configuration Locked* to *Lock.*; changed the meaning for the value *Any other value* from *No Change* to *Do nothing.*
- In the Transient Configuration Lock (FCCU_TRANS_LOCK) register description, changed the register name from *Transient Lock* to *Transient Configuration Lock*; in the general description, changed *The FCCU_TRANS register is used for unlocking configuration by writing BCh. This register is available only in supervisor mode. This register is write only.* to *Writable only by code running in Supervisor mode. Temporarily locks and unlocks the configuration. Locking the configuration prevents FCCU from entering Configuration state.*; changed the access type of the reserved field from Read-Only Zero to Reserved, Unimplemented. In the TRANSKEY field description, changed *Transition Unlocking Key value* to *Transient Configuration Lock—Writable only by code running in Supervisor mode. Temporarily locks and unlocks the configuration. Locking the configuration prevents FCCU from entering Configuration state.*; changed the meaning for the value BCh from *Configuration Unlocked* to *Unlock.*; changed the meaning for the value *Any other value* from *FCCU gets transiently locked again.* to *Lock.*

- Because the fault-output (EOUT) signaling frequency is programmable (FOP), not fixed at 61 Hz:
  - In EOUT interface, deleted *With a CLKSAFE frequency of 16 MHz, this drives a signal of 61 Hz on EOUT.*
  - In Dual-rail protocol, deleted *As long as FCCU is in NORMAL or ALARM state, output will show "non-faulty" signal. Output pins EOUT[0:1] will toggle between 01 and 10 with a given frequency of 61 Hz.* Note that, although deleted here, the information regarding the "nonfaulty" indication on the EOUT signals during Normal and Alarm states remains true and is documented elsewhere.

- In the Configuration (FCCU_CFG) register description, in the FCCU_SET_CLEAR field description, added the field name, Fault-Output (EOUT) Control; rewrote the function description and value meanings.

- In the FCCU_SET_AFTER_RESET field description, added the spelled-out name of the field, Fault-Output (EOUT) Activate; rewrote the general description; rewrote the meanings for both values.

- In the Delta T (FCCU_DELTA_T) register description, in the DELTA_T field description, added the spelled-out name of the field, Bistable Minimum Fault-Output (EOUT) Faulty Interval; added the following sentence: *Applies only to Bistable fault-output mode (FOM)*; changed *EOUT fault-indication timer* to *fault-output (EOUT) timer*; moved the information that is after the equation to EOUT interface because it's general and not only applicable to DELTA_T.
- Inserted the following new sections in EOUT interface prior to Dual-rail protocol :
  - Introduction. Moved the first paragraph from the previous EOUT interface section into this section. In the moved information, replaced *The FCCU provides up to two bidirectional signals (EOUT[1:0]) as a failure indication to the external world.* with *You use the EOUT[1:0] signals to indicate FCCU's condition to off-chip logic.*
  - The FCCU conditions. Added new content.
  - How the fault-output (EOUT) timer works. Moved general information from the DELTA_T field description to this section. In the moved information, changed *EOUT fault-indication timer* to *fault-output (EOUT) timer*. In the first paragraph of the moved information, added the following new list item to the beginning of the list: *The fault-output (EOUT) signals are not programmed to be always low (FCCU_SET_CLEAR)*. In the third paragraph of the moved information, changed the first list item from *T_min has expired* to *If the EOUT signals are in Bistable fault-output mode (FOM), T_min has expired*. In the second paragraph, deleted *or not applicable*.
  - Prepare the EOUT signals to indicate FCCU's condition. Added new content.
  - More about the EOUT interface. In one of the notes, changed *after software has enabled the EOUT signals (FCCU_CFG[FCCU_SET_AFTER_RESET]* to *when the EOUT signals are active (FCCU_SET_AFTER_RESET)*.
- Revised the Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*) register description for clarity.

- In Signals, rewrote the description of the EOUT[1:0] signal function for accuracy and clarity.
- In the figure in Block diagram, changed the double-headed arrow between *EOUT1 intf* and *EOUT[1]* to a single-headed arrow that indicates that EOUT[1] is an output only.

• In the Status (FCCU_STAT) register description, rewrote the ESTAT and STATUS field descriptions for clarity.
• In the IO Control (FCCU_EINOUT) register description, for the EIN1 and EIN0 fields, changed the reset value to indeterminate (x) and rewrote the field descriptions for clarity. • In Signals, added a table row for the EIN[1:0] signals.

• Added the following sections to clarify how to perform certain procedures:
  • Put FCCU in Configuration or Normal state
  • Run operations
  • Lock and unlock the configuration

• In Acronyms and abbreviations, added a row for *NVM*.
In Introduction : • Replaced the list item that begins with *The FCCU's configuration is lockable* with *Lockable configuration*. • Added the list item *Nonvolatile memory interface (NVMCFG signals) for configuration loading*.
• In Main features, added the list item *Nonvolatile memory interface (NVMCFG signals) for configuration loading*.
• In the Control Key (FCCU_CTRLK) register description, rewrote the description of the register's function and the description of the CTRLK field's function so it includes the OP31 operation.
• Added FCCU_CFG register bit value sources (N and C) by event.
• Added Configuration registers.
• In Signals, added table rows for the NVMCFG[21:0] and SSCMDONE signals.
• In the Configuration-State Timer Interval (FCCU_CFG_TO) register description, changed its name to *Configuration-State Timer Interval*, and rewrote the description of the register's function and the description of the TO field's function.
• In the Configuration (FCCU_CFG) register description, rewrote the description of the register's function and rewrote the footnote on the register diagram.

• In the Control (FCCU_CTRL) register description:
  • Rewrote the description of the register's function.
  • Rewrote the description of the OPR field.

• Added Nonvolatile memory interface.
• In Introduction, changed *They show operational state only on software request* to *They show operational state only after configuration by software*.
• In More about the EOUT interface, deleted this paragraph: *These resets influence the state of the failure indication pins…Other resets should not influence the state of the failure indication pins*.
• In the Delta T (FCCU_DELTA_T) register description, rewrote the note in the DELTA_T field description to clarify that the variance of the DELTA_T durations is chip-specific.
• In More about the EOUT interface, changed *NA (not applicable) appears in the following table* to *NA (not applicable) appears in the following table*.
• In the Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*) register description, rewrote the register's function description and the EOUTENx field's function description.
• In NMI/WKPU interface, changed *another mode* to *another mode change*.

• Throughout the chapter, changed *non-critical* to *noncritical*.
• In each of the following register descriptions, changed the format of the table in the register function summary and reworded the sentence in the field function summary that refers to the table:
  • Noncritical Fault Configuration (FCCU_NCF_CFG*n*)
  • Noncritical Fault Status (FCCU_NCF_S*n*)
  • Noncritical Fault Enable (FCCU_NCF_E*n*)
  • Noncritical Fault Timeout Enable (FCCU_NCF_TOE*n*)
  • IRQ Alarm Enable (FCCU_IRQ_ALARM_EN*n*)
  • Bistable Fault-Output (EOUT) Mode Signaling Enable (FCCU_EOUT_SIG_EN*n*) (no sentence rewording)

• In Introduction, changed *One of the error out pins is high* to *One of the fault-output (EOUT) signals is high*.

A.75 STCU2 changes

A.75.1 STCU2 configuration changes

• In On-Line reset generation (only MBIST) added a Note "During online selftest execution, ... will generate reset."
• Added Supported BIST sequence section. • Updated section AUTOLOCK_VALUE for register write access and DCF_COMPLETION value for DCF write completion.
• Added note "A functional reset must be generated for proper functioning of the chip after online MBIST self-test" to On-Line reset generation (only MBIST)

A.75.2 STCU2 module changes

• In the STCU2 Run Software Register (STCU2_RUNSW) register description, changed the access type of bit 20 from read-write to read-only zero.

• In the Register write-access watchdog timer section:
  • Revised the first paragraph and reorganized part of its content into a list.
  • In the second list item, added a sentence: "To refresh the hardware watchdog timer before it times out, write only security key 2 to STCU2_SKC."
  • In the Note:
  • Replaced "8 STCU2 clock cycles" with "a number of STCU2 clock cycles equal to DCF_COMPLETION."
  • Replaced "AUTOLOCK Value divided by 8" with "AUTOLOCK_VALUE divided by DCF_COMPLETION."
  • For DCF_COMPLETION, added a reference to the chip-specific STCU2 information.
  • Removed a sentence: "To refresh the hardware watchdog timer, the user must write key 2 into the STCU2_SKC register."
  • Edited the last paragraph.

• In the second-to-last paragraph of the STCU2 SK Code Register (STCU2_SKC) register description, changed *a transfer error on the IPS or SSCM bus* to *a transfer error on the IPS bus*.

• Added the following sections to clarify terminology:
  • BISTs and BIST partitions
  • BIST sequences

• In the STCU2 Run Software Register (STCU2_RUNSW) register description, fixed bit 21 in the diagram.
• In the register description, fixed bit 31 in the diagram.

• STCU2 Watchdog Granularity (STCU2_WDG) :
  • Throughout: Edited for clarity
  • Note added to WDG[WDGEOC] field description: "If this register is programmed while STCU2_RUN[RUN]=1, the new STCU2_WDG value, programmed after STCU2_RUN, does not take effect."

• Offline self-test sequence :   • Removed: "Optionally, also the STCU2_LB_PRPGL/H can be programmed in order to load a specific seed."
• In the STCU2 Off-Line MBIST Status Medium Register (STCU2_MBSM) register description, in the note in each of the field descriptions, changed *STCU2_MBEL* to *STCU2_MBEM*.

• Online self test sequence :
  • New step added: "Ensure the RUN_CFG field value in all implemented MC_ME_PCTL*n* registers forces the associated peripheral to a frozen, clock gated state."
  • Numerous editorial changes

• Steps changed to a numbered sequence instead of a bulleted list • STCU2 Register names referenced in this section are no longer prefixed with "STCU2_".
• In the STCU2 Configuration Register (STCU2_CFG) register description, changed *FSL production* to *MCU production*.

A.76 CRC module changes

• In Input Register (CRC_INP), added a Note.
• Replaced "Ethernet" instances with "ITU-T V.42" standard throughout the chapter • In Standard features, updated the Note
• In Configuration Register (CRC_CFG), updated reset value footnote for POLYG field.

A.77 REG PROT changes

A.77.1 REG_PROT configuration changes

• In section, Mode Entry (MC_ME) protected registers, table: Protected MC_ME registers: • Removed information pertaining to ME_PCTL81

A.77.2 REG_PROT module changes

• In Global Configuration Register (REG_PROT_GCR)
  • Removed Note "Access in....read/write" from register description.
  • Modified Note for HLB bit as "Access in....access is read/write." and its description.
  • Added description to UAA bit as "Controls the User and....REG_PROT_SLBR*n*."

A.78 PASS changes

A.78.1 PASS configuration changes

• Removed the attachment *PASS_Lock_register_details* spreadsheet from RM and updated all the relevant details in this document by adding the following sections:
  • PASS_LOCK0_PGn register bit mapping
  • PASS_LOCK1_PGn register bit mapping
  • PASS_LOCK2_PGn register bit mapping
  • PASS_LOCK3_PGn register bit mapping
  • Debug Interface Access
  • Flash Test Mode protection

- In PASS_LOCK3_PGn register bit mapping FSL_PROD changed to MCU_PROD.

A.78.2 PASS module changes

- In register: Life Cycle Status Register (PASS_LCSTAT) :
  - Removed reset value and added footnote 'Reset value is the Life Cycle ... ' to bit 'LIFE'.
- In register bit field: LIFE :
  - Changed 'Freescale Production' to 'MCU Production'
- In section Test mode enable password,Test mode enable password generation
  - Changed 'FSL' to 'Manufacturer'
- Editorial change: Added cross ref in Censorship

A.79 TDM changes

A.79.1 TDM configuration changes

- No substantial content changes

A.79.2 TDM module changes

- In section Flash erase counter and tamper detection
  - Removed text "Alternatively,...the diary length for that TDR." from second note in this section.

A.80 Flash Memory Programming and Configuration changes

- Added new Chapter

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTest, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and µVision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2016 NXP B.V.

Document Number MPC5746CRM
Revision 4, 12/2016



